



SCMS SCHOOL OF ENGINEERING & TECHNOLOGY

VIDYA NAGAR, KARUKUTTY, ERNAKULAM – 683576, PHONE: 0484-2882900, 2450330

E-Mail: sset@scmsgroup.org Website: www.scmsgroup.org/sset

1.1.1: The Institution ensures effective curriculum delivery through a well-planned and documented process

- The College ensures effective curriculum delivery through a well-planned and documented process. In curriculum related activities, academic freedom is enjoyed by the faculty. They actively participate and support the curriculum development, delivery, and assessment within the overall framework of the program approved by the University.
- The institution follows the academic curriculum designed by A P J Abdul Kalam Technological University (APJAKTU). Flexibility in making changes in curriculum is restricted due to this. But the curriculum provided by the university has got a balanced structure comprising humanities, basic science, professional core and elective courses. In 2019 the curriculum had been revised after the first introduction of curriculum in 2015 by APJAKTU. The University has taken into consideration the Outcome Based Education Parameters and the curriculum has been prepared such that it helps in the attainment of Programme Outcomes to a large extent. At the same time within the institution, efforts are taken to assess the curriculum and syllabus and additional measures are taken to help attainment of PO's and PSO's.
- The academic calendar is published by the university every year before the commencement of the classes. The college plans the academic schedule as per APJAKTU's academic calendar. Departmental academic calendar and institutional academic calendars are prepared based on this. It details the academic activities for the semester, and provides schedule for conduct of class/course committee meetings, continuous internal evaluations as well as commencement date of end semester examinations. The institutional academic calendar is published in the college website.
- Course allotment is done well in advance, for faculty to prepare course plan, and for effective planning and preparation by the faculty for fruitful curriculum delivery. Before the commencement of the semester a detailed course plan has to be prepared for each course by the concerned faculty and duly approved by the Head of the Department and the Principal. The concerned teachers prepare their subject-wise lesson plans. The lesson plan incorporates topics to be covered and, the number of hours needed for completing each topic.
- The timetable committee prepares time table in adherence to the requirements specified by APJAKTU based on which teachers conduct classes. For ensuring adherence to academic calendar, course diary is prepared and maintained for each course by the respective faculty.
- Meetings are arranged periodically to review the coverage of syllabus in the respective departments and suitable corrective measures are adopted to complete the syllabus within the stipulated time. Bimonthly meetings are conducted by the academic committee to monitor the progress of the completion of syllabus. The adherence to academic calendar is also verified in course committee/class committee meetings and during academic audits done internally and externally. Internal academic auditing committee monitors the adherence to academic calendar and all the academic activities.



SCMS SCHOOL OF ENGINEERING & TECHNOLOGY

VIDYA NAGAR, KARUKUTTY, ERNAKULAM – 683576, PHONE: 0484-2882900, 2450330

E-Mail: sset@scmsgroup.org Website: www.scmsgroup.org/sset

This is followed by an external audit by the University. ISO audit team also ensures the quality of the teaching learning process.

- Continuous Internal Assessments (CIE) are conducted in accordance with the university academic calendar. For the effective transmission and delivery of curricula, departments integrate classroom teaching with various ICT tools, laboratory practical, field projects, students' seminars, tutorials, question papers solving etc.
- Special classes and remedial classes are conducted for students lagging in understanding concepts and to bring them at par with the rest of the class. Bright students are encouraged and assisted to improve upon the academic grades. For enhancing subject knowledge invited talks, workshops, seminars, conferences etc. are organised.
- The classes were at the height of their activity when the temporary closure was announced by Government in March 2019, due to the outbreak of the pandemic demanding social distancing, The U.G. and P.G. courses were in the middle of even semester. After the initial chaos of few days, SSET resumed the teaching/learning process remotely. The announcement was made through the official website and communicated with the students and parents through the faculty advisors. As the initial step towards remote learning, it was decided to create digital learning resources, both textual and visual, for the course contents to continue with the classes. These course contents were communicated to the students through Google Classroom. IQAC took care of the whole procedure, to ensure all courses, theory and laboratory are assigned with Google Classroom and the entire students of the respective classes can access the learning resources.
- SSET became a part of the Coursera for Campus Campaign. In addition to Coursera, SSET encouraged students to do NPTEL courses as well. A good number of webinars, workshops, and talks have been conducted by all departments, where students got a medium to interact with the supporters from Academia and Industry. Students were also encouraged to be part of online- hackathons and other peer-driven activities to interact with the outside world. Project guidance and evaluations, remedial classes, and mentoring sessions were also conducted through the online conferencing platforms. Whenever University & Government orders permitted special classes and laboratory sessions were held offline in 2021.
- The offline classes resumed in full swing with classes for final years, i.e., for seventh semester students from 25.10.2021. This was followed by commencement of classes for all semesters UG and PG and by 22.11.2021 the academic activities were fully resumed in offline mode and wherever required hybrid mode was adopted.



SCMS SCHOOL OF ENGINEERING & TECHNOLOGY

VIDYA NAGAR, KARUKUTTY, ERNAKULAM – 683576, PHONE: 0484-2882900, 2450330

E-Mail: sset@scmsgroup.org Website: www.scmsgroup.org/sset

Appended:

- ACADEMIC CALENDAR
- TIME TABLE
- SAMPLE COURSE FILE



A handwritten signature in green ink, appearing to be 'Dr. Praveensal C.J.'.

DR. PRAVEENSAL C.J.
PRINCIPAL
SCMS SCHOOL OF ENGINEERING & TECHNOLOGY



APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY

Academic Calendar July 2017– July 2018

(B.Tech, B.Arch, M.Tech, M.Arch, M.Planning, MCA and Evening B.Tech&M.Tech)

Please see separate Academic Calendar for MBA

Day	July 2017	August 2017	September 2017
Mon			
Tue		1	Commencement of Classes, Registration Starts
Wed		2	
Thu		3	
Fri		4	1 Onam Vacation Begins Bakrid
Sat		5	2
Sun	2	6	3 1 st Onam
Mon	3	7	4 Course Committee/Class Committee Meeting
Tue	4	8	5 3 rd Onam
Wed	5	9	6 SreeNarayana Guru Jayanthi
Thu	6	10	7
Fri	7	11	8
Sat	8	12	9
Sun	9	13	10
Mon	10	14	11 Re-Opening
Tue	11	15	12 Independence Day Sreekrishna Jayanthy
Wed	12	16	13 Registration Ends Publish Attendance
Thu	13	17	14
Fri	14	18	15
Sat	15	19	16 Test I to be Completed
Sun	16	20	17
Mon	17	21	18 Commencement Engineering handholding program for new entrant.
Tue	18	22	19
Wed	19	23	20
Thu	20	24	21 SreeNarayana Guru Samadhi Day
Fri	21	25	22 Publish Test I Marks
Sat	22	26	23
Sun	23	27	24 KarkadakaVaavu
Mon	24	28	25 Birthday of Ayyankali
MTu	25	29	26
Wed	26	30	27
Thu	27	31	28
Fri	28		29 Mahanavami
Sat	29		30 Vijayadasami , Muharram
Sun	30		
Mon	31		



APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY

Academic Calendar July 2017– July 2018

(B.Tech, B.Arch, M.Tech, M.Arch, M.Planning, MCA and Evening B.Tech&M.Tech)

Please see separate Academic Calendar for MBA

Day	October 2017	November 2017	December 2017
Mon			
Tue			
Wed		1	
Thu		2	
Fri		3	1
Sat		4	2
Sun	1	5	3
Mon	2	6	4
Tue	3	7	5
Wed	4	8	6
Thu	5	9	7
Fri	6	10	8
Sat	7	11	9
Sun	8	12	10
Mon	9	13	11
Tue	10	14	12
Wed	11	15	13
Thu	12	16	14
Fri	13	17	15
Sat	14	18	16
Sun	15	19	17
Mon	16	20	18
Tue	17	21	19
Wed	18	22	20
Thu	19	23	21
Fri	20	24	22
Sat	21	25	23
Sun	22	26	24
Mon	23	27	25
Tue	24	28	26
Wed	25	29	27
Thu	26	30	28
Fri	27		29
Sat	28		30
Sun	29		31
Mon	30		
Tue	31		



APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY

Academic Calendar July 2017– July 2018

(B.Tech, B.Arch, M.Tech, M.Arch, M.Planning, MCA and Evening B.Tech&M.Tech)
Please see separate Academic Calendar for MBA

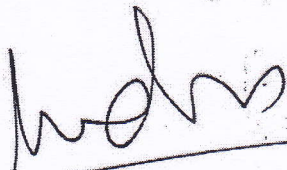
Day	January 2018	February 2018	March 2018
Mon	1 Registration Starts Commencement of Even Semester Classes		
Tue	2 Mannam Jayanthi		
Wed	3		
Thu	4	1	1
Fri	5	2 Publish Attendance	2 Publish Attendance
Sat	6	3	3
Sun	7	4	4
Mon	8 Course Committee/Class Committee Meeting	5	5
Tue	9	6	6
Wed	10	7	7
Thu	11	8	8
Fri	12 Registration Ends	9	9
Sat	13	10 Text 1 to be Completed	10 Test 2 to be Completed
Sun	14	11	11
Mon	15	12	12
Tue	16	13 Maha Shivratri	13
Wed	17	14 Publish Test 1 Marks	14
Thu	18	15	15
Fri	19	16 Tech Fest	16 Publish Test 2 Marks
Sat	20	17	17
Sun	21	18	18
Mon	22	19	19
Tue	23	20	20
Wed	24	21	21
Thu	25	22	22
Fri	26 Republic Day	23	23
Sat	27	24	24
Sun	28	25	25
Mon	29	26	26
Tue	30	27	27
Wed	31	28	28
Thu			29 Maundy Thursday
Fri			30 Good Friday
Sat			31
Sun			
Mon			
Tue			

SCMS SCHOOL OF ENGINEERING AND TECHNOLOGY, KARUKUTTY

Schedule of Activities

July 2017- December 2017

21-07-2017	: Yoga Day
27-07-2017	: Inauguration of 17 th B-Tech Batch
26-08-2017	: Onam celebration
11 th -14 th Sept. 2017	: Internal Test 1
3 rd to 7 th Oct. 2017	: Open House Meeting
23-10-2017	: Agneya 2017
31 st , Oct., 1 st &2 nd Nov. 2016	: Internal Test 2
17-11-2017	: PPTIA 2017Curtain Raiser
29-11-2017	: Blood Donation
7 th - 12 th Dec. 2017	: SSET Staff Badminton Tournament
22-12-2017	: Christmas Celebration


Principal

25/6/2017

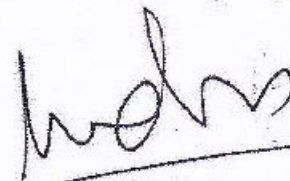
SCMS SCHOOL OF ENGINEERING AND TECHNOLOGY, KARUKUTTY

Schedule of Activities

January 2018 - June 2018

2 nd and 3 rd Feb 2018	:Dr. PPTIA 2017
16-02-2018	:IEE Ascendio
7 th -9 th Feb 2018	:Internal exam 1 (S ₂ ,S ₄ ,S ₆)
26 th -28 th Feb 2018	:Internal exam 1 (S ₈)
02-03-2018	:Blood Donation
12 th to 20 th March 2018	:Dr. PPT Memorial all Kerala Volley ball Tournament (Mens & Womens)
19 th ,21 st &23 rd March 2018	:Internal exam 2 (S ₂ ,S ₄ ,S ₆ ,S ₈)
24 th ,26 th & 28 th	:Model Exam (S ₁)
6 th and 7 th April 2018	:Igniz 2017
4 th June 2018	:World Environment Day
20 th June 2018	:Dr. PPTIA 2018 Curtain Raiser

18/12/2017



Principal

SCMS SCHOOL OF ENGINEERING AND TECHNOLOGY
Vidya Nagar, Karukutty - 683 582

Time Table w.e.f 01-01-2018

COMPUTER SCIENCE & ENGINEERING -2 (S2)

	1	2	3		4	5	6	7
Day	9:15 to 10.05	10:05 to 10:55	11:05 to 12:00	L U N C H B R E A K	12.45 to 1.35	1.35 to 2.25	2.35 to 3:25	3.25 to 4.20
Mon.	CP	EG	DE		BCE	PHY	MATHS	DE
Tue	PHY	MATHS	CP		BCE	EG		CP
Wed	MATHS	CP	DE		BCE	CP/CE(BO/MNG*)		
Thu	BCE	PHY	DE		PHY	MATHS	EG	
Fri	PHY/CP/CE (STV*/BO/MNG)				EG	CP	PHY	ENG

<u>Subject</u>		<u>Name of Faculty</u>
MATHS	Calculus	MT Mini Tom
PHY	Engg. Physics	STV Sruthi T V
EG	Engineering Graphics	VP Dr. Venu P
DE	Design & Engineering	BRN Balu R Nair
CP	Computer Programming	BO Bini Omman
BCE	Basics of Civil Engineering	CM Chelsa Mariam
ENG	English	JT Jane Theresa

Saturday's Time table will be on a rotation basis (Monday to Friday) and will be intimated well in advance.



PRINCIPAL

SCMS SCHOOL OF ENGINEERING AND TECHNOLOGY

Vidya Nagar, Karukutty - 683 582

Time Table w.e.f 01-08-2019

ELECTRONICS & COMMUNICATION ENGINEERING (S3)

	1	2	3	4		5	6	7
Day	8.45-9.40	9.40-10.35	10.45-11.35	11.35-12.25		1.10 - 2.05	2.15 - 3.05	3.05 - 4.00
Mon.	LCD	EC	LACA	SSD	L U N C H B R E A K	EDA LAB		
Tue	EC	NT	LACA	BE		NT	SSD	LCD
Wed	NT	LACA	SSD	EC		LCD	LACA (T)	EC
Thu	LCD	EC	SSD	NT		BE	LACA	SSD(T)
Fri	LCD	EDC LAB				BE	NT(T)	EC(T)

	<u>Subject Code</u>	<u>Subject</u>		<u>Name of Faculty</u>
Maths	MA201	Linear Algebra & Complex Analysis	SC	Sophia Cleetus
NT	EC201	Network theory	PNP	Prathibha N Pillai
SSD	EC203	Solid State Devices	PM	Parvathy M.
EC	EC205	Electronic Circuits	DB	Deepa B.
LCD	EC207	Logic Circuit Design	JK	Jerry Kuriakose
BE	HS200	Business Economics	SM	Subbaiyya M
EDC LAB	EC231	Electronic Devices & Circuits Lab	UN	Deepa B*, Uma N.
LAB	EC233	Electronic Design Automation Lab	V*,SL	Anandhi V*, Srilekshmi M.



PRINCIPAL

SCMS SCHOOL OF ENGINEERING AND TECHNOLOGY

Vidya Nagar, Karukutty - 683 576

Time Table w.e.f. 22-03-2021

ELECTRICAL & ELECTRONICS ENGINEERING (S6)

	1	2	3	LUNCH BREAK	4	5 (OFFLINE)
Day	9.00 to 10am	10.10 to 11.10am	11.20-12.20pm		1.20pm to 2.20pm	2.30pm-3.30pm
Mon.	EMT	ACT	PSA		S & C LAB	POM
Tue	ACT	POM	EMT		PSA	ED
Wed	PSA	ED	ACT		PED LAB	BMI
Thu	ED	BMI	POM		EMT	ACT
Fri	EMT	PSA	BMI		ACT	PSA
Sat	BMI	ED	POM		CE	EMT

	Course Code	Course Name	Name of Faculty		Google Classroom code
EMT	EE 302	Electromagnetics	DNK	Dr. Divya Nath K	vvjk2ri
ACT	EE 304	Advanced Control Theory	LCR	Lakshmi C.R	5zgbolt
PSA	EE 306	Power System Analysis	JS	Jayalakshmi S	uh4ckv7
ED	EE 308	Electric Drives	NKM	Dr. Nandakumar M	ijcnmux
POM	HS 300	Principles of Management	AVT	Anjana Vishwanath	3nkck5q
LECTIVE	EE 372	Biomedical Instrumentation	VJ	Varun Jose	zgtvrrn
S&C LAB	EE332	Systems & Control Lab	DNK*, PV	Dr. Divya Nath K & Priya Venugopal	ou5vlre
PED LAB	EE334	Power Electronics & Drives Lab	LB* & JS	Lekshmi Banu & Jayalakshmi S	hp2pzm4
CE	EE352	Comprehensive Exam	JS * & BP	Jayalakshmi S Beena Puthillath	k2xagc3



PRINCIPAL

SCMS SCHOOL OF ENGINEERING AND TECHNOLOGY
VIDYA NAGAR, KARUKUTTY, ERNAKULAM - 683 582



COURSE DIARY
THEORY

Programme..... *B.Tech*

Name of Faculty	<i>Bini Omman</i>	
Mobile No.	<i>8281371594</i>	
Designation & Department	<i>Asst. Professor in CSE</i>	
Course Code	<i>CS100</i>	
Course Name	<i>Computer Programming</i>	
Semester & Year	<i>2nd Semester 2018</i>	
Branch & Batch	<i>CSE, Batch 17</i>	
Semester Duration	<i>From: 1st January 2018</i>	<i>To: 12th April 2018</i>

SCHEDULE OF WORK

Day	1	2	3	4	5	6	7
Monday	CP						
Tuesday			CP				CP
Wednesday		CP					
Thursday							
Friday					CP		

GENERAL INSTRUCTIONS

- Student performance should be evaluated solely on an academic basis.
 - Student's evaluation should be fair, consistent, transparent and accountable.
 - Evaluation of student's performance should be disclosed to the students.
1. Keep the Course Diary up to date by clearly indicating the subject coverage and students attendance on the relevant pages.
 2. Paste the syllabus in the relevant page.
 3. Write / Paste the Course plan in the relevant page.
 4. Events in a semester such as Series Test Days, Cultural / Celebration Days, days for extra co-curricular activities etc. may be indicated in the Year Calendar.
 5. Assignment details may be written in the Course Diary or may be filed in the Course File.
 - (i) Minimum 3 Nos. of assignments should be given.
 - (ii) Different sets of questions may be given in an assignment (atleast three) to a class.
 - (iii) Assignments may be in the form of written - closed / open book, individual / group, home assignment, or in the form of oral presentation, quiz, seminar etc.
 6. Show complete split up of sessional marks in the page "Particulars of Marks". Final sessional mark for each student should be equal to the sum of marks awarded for Assignments (10) and Class Tests (40).
 7. All the entries in the course diary must be, legibly written without overwriting and free of errors.
 8. The staff member will be responsible for the safe custody of the Course Diary and (s) he should return it to the HOD at the end of semester or earlier if (s) he leaves the department or discontinue the subject.
 9. Follow KTU regulations for computing sessional marks.

PRINCIPAL

Course No.	Course Name	L-T-P-Credits	Year of Introduction
CS100	Computer Programming	2-1-0	2016

Course Objectives

To understand the fundamental concept of C programming and use it in problem solving.

Syllabus

Introduction to C language; Operators and expressions; Sorting and searching; Pointers; Memory allocation; Stacks and Queues.

Course Outcomes

1. Identify appropriate C language constructs to solve problems.
2. Analyze problems, identify subtasks and implement them as functions/procedures.
3. Implement algorithms using efficient C-programming techniques.
4. Explain the concept of file system for handling data storage and apply it for solving problems
5. Apply sorting & searching techniques to solve application programs.

References

1. Rajaraman V., Computer Basics and Programming in C, PHI.
2. Anita Goel and Ajay Mittal, Computer fundamentals and Programming in C., Pearson.
3. Gottfried B.S., Programming with C, Schaum Series, Tata McGraw Hill.
4. Horowitz and Sahni, Fundamentals of data structures - Computer Science Press.
5. Gary J. Bronson, ANSI C Programming, CENGAGE Learning India.
6. Stewart Venit and Elizabeth Drake, Prelude to Programming – Concepts & Design, Pearson.
7. Dromy R.G., How to Solve it by Computer, Pearson.
8. Kernighan and Ritchie D.M., The C. Programming Language, PHI.

COURSE PLAN

Module	Contents	Contact Hours	Sem.Exam Marks;%
I	Introduction to C Language: Preprocessor directives, header files, data types and qualifiers. Operators and expressions. Data input and output, control statements.	7	15%

II	Arrays and strings- example programs. Two dimensional arrays - matrix operations. Structure, union and enumerated data type.	8	15%
III	Pointers: Array of pointers, structures and pointers. Example programs using pointers and structures.	7	15%
FIRST INTERNAL EXAM			
IV	Functions – function definition and function prototype. Function call by value and call by reference. Pointer to a function –. Recursive functions.	7	15%
SECOND INTERNAL EXAM			
V	Sorting and Searching : Bubble sort, Selection sort, Linear Search and Binary search. Scope rules Storage classes. Bit-wise operations.	6	20%
VI	Data files – formatted, unformatted and text files. Command line arguments – examples.	7	20%
END SEMESTER EXAM			

2014



SCMS SCHOOL OF
ENGINEERING & TECHNOLOGY, KARUKUTTY

COURSE INFORMATION & COURSE PLAN

5. CS100

Computer Programming



5.1 COURSE INFORMATION

Course Syllabus

Course Code	Course Name	L-T-P-Credits	Year of Introduction
CS100	COMPUTER PROGRAMMING	2-1-0-3	2016
Prerequisites: NIL			
Course objectives: <ul style="list-style-type: none">• To impart knowledge about programming in C• To learn basics of PYTHON			
Syllabus: Introduction to Programming, Basic elements of C, Control statements in C, Arrays and Strings, Functions, Storage classes, Structures and Pointers, File Management in C, Introduction to PYTHON			
Expected outcome: <ol style="list-style-type: none">1. Ability to design programs using C language2. Ability to develop simple programs using PYTHON			
Text Books: T1. E. Balaguruswamy, <i>Programming in ANSI C</i> , Tata McGraw Hill, New Delhi T2. John V Guttag, <i>Introduction to Computation and Programming using PYTHON</i> , PHI			
References: R1. Norton, <i>Peter Norton's Introduction to Computers</i> , Tata McGraw Hill, New Delhi R2. Byron S. Gottfried, <i>Programming with C</i> , Schaun Outlines-McGraw Hill R3. Ashok Kamthane, <i>Programming with ANSI & Turbo C</i> , Pearson Education R4. K.R Venugopal and S.R Prasad, <i>Mastering C</i> , Tata McGraw Hill R5. Kelly, Al & Pohl, <i>A Book on C-Programming in C</i> , 4 th Ed., Pearson Education			



--

COURSE PLAN			
Module	Contents	Hours	Sem. Exam Marks %



I	Introduction to Programming: Machine language, assembly language, and high level language, Compilers and assemblers. Flow chart and algorithm – Development of algorithms for simple problems Basic elements of C: Structure of C program – Keywords, identifiers, data types, operators and expressions, Input and Output functions	5	15
II	Control statements in C: if, if-else, while, do-while and for statements, switch, break, continue, go to, and labels, Programming examples	7	15
FIRST INTERNAL EXAMINATION			
III	Arrays and Strings: Declaration, initialization, processing arrays and strings – two-dimensional and multi-dimensional arrays – application of arrays, Example programs	7	15
IV	Functions: Functions – declaring, defining and accessing functions – parameter passing methods – passing arrays to functions, Recursion	7	15
SECOND INTERNAL EXAMINATION			
V	Structures: declaration, definition and initialization of structures, unions Pointers: Concepts, declaration, initialization of pointer variables, Accessing a Variable through its Pointer Chain of Pointers, Pointer Expressions, Pointer Increments and Scale Factor, Pointers and Arrays, Examples	8	20
VI	File Management: File operations, Input/Output Operations on Files, Random Access to Files, File pointer Introduction to PYTHON: Basic Syntax, Operators, control statements, functions-examples	8	20



QUESTION PAPER PATTERN (End Semester Examination)

Part A: 8 questions

One question from each module of Module I-IV; and two each from Module V & VI. Student has to answer all questions: $(8 \times 5) = 40$

Part B: 3 questions uniformly covering modules I & II

Student has to answer any 2 questions: $(2 \times 10) = 20$

Part C: 3 questions uniformly covering modules III & IV

Student has to answer any 2 questions: $(2 \times 10) = 20$

Part D: 3 questions uniformly covering modules V & VI

Student has to answer any 2 questions: $(2 \times 10) = 20$

Note: Each question can have maximum of 4 sub questions, if needed.

COURSE OBJECTIVES

1.	To impart knowledge about programming in C
2.	To learn basics of PYTHON

COURSE OUTCOMES

Sl. No.	Course Outcomes	Bloom's Taxonomy Level
CS100.1	Understanding about the various programming constructs used in C programming	Level 1
CS100.2	Thorough understanding of one dimensional and two dimensional arrays and will be able to use it for solving real world problems	Level 1 & Level 4
CS100.3	familiarization of pointers and its implementation	Level 3



CS100.4	Will be able to analyze problems, identify subtasks and implement it using functions.	Level 2 & Level 3
CS100.5	Familiarization of storage classes in C and will be able to apply various sorting and searching for various application programs.	Level 3
CS100.6	Understanding about the various programming constructs used in C programming	Level 3

CO-PO MAPPING

PO \ CO	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PO 12
CS100.1	2		2	2								1
CS100.2	2	2	3	2								1
CS100.3	2	2	3	2								1
CS100.4	2	2	3	2								1
CS100.5	2	2	2	2								1
CS100.6	2	2	3	2								1
CS100 (overall level)	2.00	2.0	2.7	2.0								1.0

CO-PSO MAPPING

PSO \ CO	PSO 1	PSO 2	PSO 3
CS100.1	2		
CS100.2	2	1	
CS100.3	2	2	2
CS100.4	2	2	
CS100.5	2	3	1

Note: (Applicable to CO-PO Mapping and CO-PSO Mapping as well)

- 1 – Slight (Low)
- 2 – Moderate (Medium)
- 3 – Substantial (High)
- '-' – No correlation



CS100 (overall level)	2	1	1
-----------------------------	---	---	---

JUSTIFICATIONS FOR CO-PO MAPPING

Mapping	Low/Medium/High	Justification
CS100.1 – PO2	2	Students can select appropriate C language construct while analyzing engineering problems.
CS100.1 – PO3	2	Students can develop solutions for complex engineering problems by selecting appropriate C language construct.
CS100.1 – PO4	2	Students can select appropriate C language construct for synthesis and interpretation of data.
CS100.1 – PO12	2	Students can recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.
CS100.1 – PSO1	2	Can apply knowledge of mathematics, science, engineering and computer science fundamentals to solve complex computational problems..
CS100.2 – PO1	2	Students can develop solutions for complex engineering problems with the knowledge of arrays.
CS100.2 – PO2	2	Students can analyze problems, identify subtasks and implement them as functions/procedures, while analyzing engineering problems.
CS100.2 – PO3	2	Students can develop solutions for complex engineering problems by implementing them as functions.
CS100.2 – PO4	M	Students can use functions for the design of experiments.
CS100.2 – PO5	L	Student can Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools
CS100.2 – PSO1	M	The ability to identify, analyze and design solutions for complex engineering problems in multidisciplinary



		areas by understanding the core principles and concepts of computer science and thereby engage in national grand challenges.
CS100.2 – PSO2	M	The ability to acquire programming efficiency by designing algorithms and applying standard practices in software project development to deliver quality software products meeting the demands of the industry.
CS100.2 – PSO3	L	The ability to apply the fundamentals of computer science in competitive research and to develop innovative products to meet the societal needs thereby evolving as an eminent researcher and entrepreneur.
CS100.3 – PO2	M	Students can develop algorithms leading to implementation of efficient C-programs while analyzing problems.
CS100.3 – PO3	M	Students can implement algorithms of complex engineering problems using efficient C programs.
CS100.3 – PO4	M	Students can conduct investigation of complex problems by implementing the algorithms in C language.
CS100.3 – PO5	L	Student can Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools.
CS100.3 – PSO1	M	The ability to identify, analyze and design solutions for complex engineering problems in multidisciplinary areas by understanding the core principles and concepts of computer science and thereby engage in national grand challenges.
CS100.3 – PSO2	M	The ability to acquire programming efficiency by designing algorithms and applying standard practices in software project development to deliver quality software products meeting the demands of the industry.
CS100.3 – PSO3	L	The ability to apply the fundamentals of computer science in competitive research and to develop innovative products to meet the societal needs thereby evolving as an eminent researcher and entrepreneur.



CS100.4 – PO1	2	Students can develop solutions for complex engineering problems with the knowledge of arrays.
CS100.4 – PO2	2	Students can analyze problems, identify subtasks and implement them as functions/procedures, while analyzing engineering problems.
CS100.4 – PO3	2	Students can develop solutions for complex engineering problems by implementing them as functions.
CS100.4 – PO4	2	Students can use functions for the design of experiments.
CS100.4 – PSO1	2	The ability to identify, analyze and design solutions for complex engineering problems in multidisciplinary areas by understanding the core principles and concepts of computer science and thereby engage in national grand challenges.
CS100.5 – PO1	2	Students can develop solutions for complex engineering problems by selecting appropriate sorting and searching algorithms.
CS100.5 – PO2	2	Students can select appropriate storage classes for synthesis and interpretation of data.
CS100.5 – PO3	2	Students will be able to use searching and sorting techniques for the development of solutions.
CS100.5 – PO4	2	Students can apply different searching and sorting techniques for the development solutions.
CS100.5 – PO12	2	Student can Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools.
CS100.5 – PSO1	2	The ability to identify, analyze and design solutions for complex engineering problems in multidisciplinary areas by understanding the core principles and concepts of computer science and thereby engage in national grand challenges.
CS100.5 – PSO2	2	The ability to acquire programming efficiency by designing algorithms and applying standard practices in software project development to deliver quality software products meeting the demands of the industry.



CS100.5 – PSO3	2	The ability to apply the fundamentals of computer science in competitive research and to develop innovative products to meet the societal needs thereby evolving as an eminent researcher and entrepreneur.
----------------	---	---

GAPS IN SYLLABUS

Sl. No.	Gap	Action Taken	Date – Month - Year	Resource Person with Designation	Percentage of Students	Relevance to POs	Relevance to PSOs

Proposed Actions:

INSTRUCTIONAL METHODOLOGIES

1. Chalk & Talk
2. LCD/SMART Boards
3. Student Assignments
4. Student Seminars
5. Web Resources
6. Add-on Courses

DIRECT ASSESSMENT METHODOLOGIES

1. Assignments
2. Student Seminars
3. Tests/Model Exams
4. University Examination



5. Student Lab Practice
6. Student Viva
7. Mini/Major Projects
8. Certifications
9. Add-on Courses
10. Others

INDIRECT ASSESSMENT METHODOLOGIES

1. Assessment of Course Outcomes (By Feedback, Once)
2. Student Feedback on Faculty (Twice)
3. Assessment of Mini/Major Projects by External Experts
4. Others

CONTENT BEYOND SYLLABUS

1. Introduction to Embedded C
 - Difference between C and Embedded C
 - Programming style
 - Basic structure of C program
2. Arduino Programming
 - Program notation: variables, functions, control flow, Arduino conventions
 - The concept of a program variable
 - Numerical values and basic numerical operators
 - if/then/else
 - Iteration using for loops



- Real world timing and the delay() function

E-CONTENT (Additional learning)

1. <https://www.programiz.com/c-programming>
Learn C (Introduction and Tutorials to C programming) – Programiz
2. <https://nptel.ac.in/courses/106104128/>
Introduction to Programming in C (video lecture series coordinated by IIT Kanpur)
3. <https://nptel.ac.in/courses/115104095/>
Computational Science and Engineering using PYTHON (video lecture series coordinated by IIT Kanpur)
4. <https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-00-introduction-to-computer-science-and-programming-fall-2008/video-lectures/>
Introduction to Computer Science and Programming – MITOPENCOURSEWARE, Massachusetts Institute of Technology

5.2 SESSION PLAN

<i>Session</i>			<i>Topics</i>	<i>Chapter/Book</i>
			MODULE I (Introduction to Programming, Basic Elements of C)	
			<i>To impart knowledge about C fundamentals such as identifiers, operators</i>	
Hours as per syllabus				
Date	Period	Duration		
Aug 2	1	1 hour	Machine language, assembly language, and high level language	T1
Aug 2	2	1 hour	Compilers and assemblers	T1



Aug 2	7	1 hour	Flow chart	R1
Aug 3	1	1 hour	Algorithm – Development of algorithms for simple problems	R2
Aug 6	4	1 hour	Structure of C program - Keywords, Identifiers, data types, Operators and expressions	T1
Aug 6	6	1 hour	Input and Output functions	T1
Total hours planned: 5				
<i>Session</i>		<i>Topics</i>		<i>Chapter/Book</i>
		MODULE II (Control statements in C)		
		<i>To impart idea about different branching statements and looping statements</i>		
Hours as per syllabus				
Date	Period	Duration		
Sep 3, Sep 6	4, 6, 7, 1	4 hours	if, if-else, while, do-while	T2
Sep 7	1	1 hour	for statement, switch, break, continue, go to and labels	R3
Sep 10	4, 6	2 hours	Programming examples	T2
Total hours planned: 7				
<i>Session</i>		<i>Topics</i>		<i>Chapter/Book</i>
		MODULE III (Arrays and Strings)		
		<i>To develop C programs using the concepts such as arrays and strings</i>		
Hours as per syllabus				
Date	Period	Duration		



Sep 13, Sep 14	7, 1	2 hours	Declaration, initialization, processing arrays and strings	T1
Sep 17	4, 6	2 hours	Two-dimensional and multi-dimensional arrays	R4
Sep 27	7	1 hour	Application of arrays	T2
Sep 28, Sep 29	1, 4	2 hours	Programming examples	T1
Total hours planned: 7				
Session		Topics		Chapter/Book
		MODULE IV (Functions)		
		<i>To learn to develop modular C programs using functions</i>		
Hours as per syllabus				
Date	Period	Duration		
Oct 1	4, 6	2 hours	Declaring, defining, and accessing functions	T1
Oct 4,	7	1 hour	Parameter passing methods – passing arrays to functions	T2
Oct 5	1	1hour	Recursion, Storage classes – extern, auto, register and static	R5
Oct 6, Oct 8, Oct 11	2, 4, 6, 7	4 hours	Example programs	R4
Total hours planned: 7				
Session		Topics		Chapter/Book
		MODULE V		



			(Structures, Pointers)	
			<i>To impart knowledge about structures, unions, and pointers</i>	
Hours as per syllabus				
Date	Period	Duration		
Oct 12, Oct 15, Oct 20	1, 4, 6, 2	4 hours	Structures: declaration, definition and initialization of structures, unions Pointers: Concepts, declaration, initialization of pointer variables	T1
Oct 22	4, 6	2 hours	Accessing a Variable through its Pointer Chain of Pointers, Pointer Expressions	T1
Oct 25	7	1 hour	Pointer Increments and Scale Factor, Pointers and Arrays	R2
Oct 26	1	1 hour	Examples	R2
Total hours planned: 8				
Session		Topics		Chapter/Book
		MODULE VI (File Management, Introduction to PYTHON)		
		<i>To have a clear idea of different file operations and to learn to develop C programs performing file operations, To familiarize the basic concepts of PYTHON</i>		
Hours as per syllabus				
Date	Period	Duration		
Oct 29	4, 6	2 hr	File Management – File operations	T2
Nov 1, Nov 2	7, 1	2 hr	Input/Output Operations on Files	T2
Nov 5,	4, 6, 7	3 hr	Random Access to Files, File pointer	T2



Nov 8				
Nov 9	1	1 hr	Introduction to PYTHON: Basic Syntax, Operators, control statements, functions – examples	T2
Total hours planned: 8				

5.3 TUTORIAL QUESTIONS

Module 1

1. Write a code to print the words “Hello World”.



```
#include <stdio.h>

int main () {

printf (“Hello, World! \n”);

return 0;

}
```

2. Write a code where variables have been declared at the top, but they have been defined and initialized inside the main function.

```
#include <stdio.h>

// Variable declaration:
extern int a, b;
extern int c;
extern float f;

int main () {

/* variable definition: */
int a, b;
int c;
```



```
float f;

/* actual initialization */
a = 10;
b = 20;

c = a + b;
printf("value of c : %d \n", c);

f = 70.0/3.0;
printf("value of f : %f \n", f);

return 0;
}
```

3. How do you implement 'const' in a code?

```
#include <stdio.h>

int main() {
    const int LENGTH = 10;
    const int WIDTH = 5;
    const char NEWLINE = '\n';
    int area;
    area = LENGTH * WIDTH;
```




```
printf("value of area : %d", area);  
printf("%c", NEWLINE);  
  
return 0;  
}
```

Module 2

4. Write a function to return the maximum between two numbers.

```
/* function returning the max between two numbers */  
int max(int num1, int num2) {  
  
    /* local variable declaration */  
    int result;  
  
    if (num1 > num2)  
        result = num1;  
    else  
        result = num2;  
  
    return result;  
}
```

5. How do you call a function? Explain with a code.



```
#include <stdio.h>

/* function declaration */
int max(int num1, int num2);

int main () {

    /* local variable definition */
    int a = 100;
    int b = 200;
    int ret;

    /* calling a function to get max value */
    ret = max(a, b);

    printf( "Max value is : %d\n", ret );

    return 0;
}

/* function returning the max between two numbers */
int max(int num1, int num2) {
    /* local variable declaration */
    int result;
```



```
if (num1 > num2)
    result = num1;
else
    result = num2;

return result;
}
```

6. Explain nesting of for loop with a code.

```
#include <stdio.h>
int main()
{
    for (int i=0; i<2; i++)
    {
        for (int j=0; j<4; j++)
        {
            printf("%d, %d\n",i ,j);
        }
    }
    return 0;
}
```

7. Write a code to print two numbers with while loop.



```
#include <stdio.h>
main()
{
int m = 5;
int n = 0;
while (m > n)
{
printf("m = %d n = %d\n",m,n );
m--;
n++;
}
}
```

8. Write a code signifying switch statement.

```
#include <stdio.h>
int main()
{
int x = 2;
switch (x)
{
case 1: printf("Choice is 1");
break;
case 2: printf("Choice is 2");
break;
```



```
case 3: printf("Choice is 3");  
break;  
default: printf("Choice other than 1, 2 and 3");  
break;  
}  
return 0;  
}
```

9. Explain goto statement with a code.

```
#include <stdio.h>  
  
#define MAX 10  
  
int main()  
{  
    int needle;  
  
    /* get input from user*/  
    printf("Please enter a number (0-10):");  
    scanf("%d",&needle);  
  
    int i;  
    for(i = 0; i < MAX;i++)  
    {
```



```
    if(i == needle)
    {
        goto end;
    }
    else
    {
        printf("Current number %d\n",i);
    }
}
printf("Loop terminated normally.");

end: printf("Jumped from the goto statement\n");

return 0;
}
```

Module 3

10. How can we declare, assign and access arrays?

```
#include <stdio.h>
```

```
#define MAX 10
```

```
int main()
```




```
{
    int needle;

    /* get input from user*/
    printf("Please enter a number (0-10):");
    scanf("%d",&needle);

    int i;
    for(i = 0; i < MAX;i++)
    {
        if(i == needle)
        {
            goto end;
        }
        else
        {
            printf("Current number %d\n",i);
        }
    }

    printf("Loop terminated normally.");

    end: printf("Jumped from the goto statement\n");

return 0;
```



```
}
```

11. Write a code to find the average of n ($n < 10$) numbers using arrays.

```
#include <stdio.h>
int main()
{
    int marks[10], i, n, sum = 0, average;
    printf("Enter n: ");
    scanf("%d", &n);
    for(i=0; i<n; ++i)
    {
        printf("Enter number%d: ",i+1);
        scanf("%d", &marks[i]);
        sum += marks[i];
    }
    average = sum/n;

    printf("Average = %d", average);

    return 0;
}
```

12. Write a C program to find the sum of two matrices of order 2*2 using multidimensional arrays.



```
#include <stdio.h>

int main()
{
    float a[2][2], b[2][2], c[2][2];
    int i, j;

    // Taking input using nested for loop
    printf("Enter elements of 1st matrix\n");
    for(i=0; i<2; ++i)
    for(j=0; j<2; ++j)
    {
        printf("Enter a%d%d: ", i+1, j+1);
        scanf("%f", &a[i][j]);
    }

    // Taking input using nested for loop
    printf("Enter elements of 2nd matrix\n");
    for(i=0; i<2; ++i)
    for(j=0; j<2; ++j)
    {
        printf("Enter b%d%d: ", i+1, j+1);
        scanf("%f", &b[i][j]);
    }

    // adding corresponding elements of two arrays
```




```
for(i=0; i<2; ++i)
for(j=0; j<2; ++j)
{
c[i][j] = a[i][j] + b[i][j];
}

// Displaying the sum
printf("\nSum Of Matrix:");

for(i=0; i<2; ++i)
for(j=0; j<2; ++j)
{
printf("%.1f\t", c[i][j]);

if(j==1)
printf("\n");
}
return 0;
}
```

Module 4

13. Write a code to display all prime numbers between two intervals.

```
#include <stdio.h>
```



```
int checkPrimeNumber(int n);
int main()
{
    int n1, n2, i, flag;

    printf("Enter two positive integers: ");
    scanf("%d %d", &n1, &n2);
    printf("Prime numbers between %d and %d are: ", n1, n2);

    for(i=n1+1; i<n2; ++i)
    {
        // i is a prime number, flag will be equal to 1
        flag = checkPrimeNumber(i);

        if(flag == 1)
            printf("%d ",i);
    }
    return 0;
}

// user-defined function to check prime number
int checkPrimeNumber(int n)
{
    int j, flag = 1;
```



```
for(j=2; j <= n/2; ++j)
{
    if (n%j == 0)
    {
        flag =0;
        break;
    }
}
return flag;
}
```

14. Write a program to find the sum of natural numbers using a recursive function.

```
#include <stdio.h>

int addNumbers(int n);

int main()
{
    int num;

    printf("Enter a positive integer: ");

    scanf("%d", &num);
```



```
printf("Sum = %d",addNumbers(num));  
  
return 0;  
  
}
```

```
int addNumbers(int n)  
  
{  
  
    if(n != 0)  
  
        return n + addNumbers(n-1);  
  
    else  
  
        return n;  
  
}
```

15. How do you pass an entire array to a function as an argument?

```
#include <stdio.h>  
void myfunc( int *var1, int var2)  
{  
    /* The pointer var1 is pointing to the first element of  
    * the array and the var2 is the size of the array. In the  
    * loop we are incrementing pointer so that it points to
```




```
* the next element of the array on each increment.  
*  
*/  
for(int x=0; x<var2; x++)  
{  
    printf("Value of var_arr[%d] is: %d \n", x, *var1);  
    /*increment pointer for next element fetch*/  
    var1++;  
}  
}  
  
int main()  
{  
    int var_arr[] = {11, 22, 33, 44, 55, 66, 77};  
    myfuncn(var_arr, 7);  
    return 0;  
}
```

Module 5

16. How do you store information of a student using structure?

```
#include <stdio.h>  
  
struct student  
{
```



```
char name[50];

int roll;

float marks;

} s;

int main()
{

printf("Enter information:\n");

printf("Enter name: ");

scanf("%s", s.name);

printf("Enter roll number: ");

scanf("%d", &s.roll);

printf("Enter marks: ");

scanf("%f", &s.marks);
```



```
printf("Displaying Information:\n");

printf("Name: ");

puts(s.name);

printf("Roll number: %d\n",s.roll);

printf("Marks: %.1f\n", s.marks);

return 0;

}
```

17. How do you declare a pointer and use it?

```
#include <stdio.h>

int main()

{
```



```
//Variable declaration
int num = 10;

//Pointer declaration
int *p;

//Assigning address of num to the pointer p
p = #

printf("Address of variable num is: %p", p);
return 0;
}
```

18. Write a program to find the sum of six numbers with arrays and pointers.

```
#include <stdio.h>
int main()
{
    int i, classes[6],sum = 0;
    printf("Enter 6 numbers:\n");
    for(i = 0; i < 6; ++i)
    {
        // (classes + i) is equivalent to &classes[i]
        scanf("%d", (classes + i));
        // *(classes + i) is equivalent to classes[i]
```



```
    sum += *(classes + i);  
}  
printf("Sum = %d", sum);  
return 0;  
}
```

Module 6

19. How do you read name and marks of students and store it in a file.

```
#include <stdio.h>  
  
int main()  
{  
    char name[50];  
    int marks, i, num;  
  
    printf("Enter number of students: ");  
    scanf("%d", &num);  
  
    FILE *fptr;  
    fptr = (fopen("C:\\student.txt", "w"));  
    if(fptr == NULL)  
    {  
        printf("Error!");  
        exit(1);  
    }  
}
```




```
for(i = 0; i < num; ++i)
{
    printf("For student%d\nEnter name: ", i+1);
    scanf("%s", name);

    printf("Enter marks: ");
    scanf("%d", &marks);

    fprintf(fptr, "\nName: %s \nMarks=%d \n", name, marks);
}

fclose(fptr);
return 0;
}
```

20. Write a python program to check whether a year is leap year or not.

```
# Python program to check if the input year is a leap year or not
```

```
year = 2000
```

```
# To get year (integer input) from the user
```



```
# year = int(input("Enter a year: "))

if (year % 4) == 0:

    if (year % 100) == 0:

        if (year % 400) == 0:

            print("{0} is a leap year".format(year))

        else:

            print("{0} is not a leap year".format(year))

    else:

        print("{0} is a leap year".format(year))

else:

    print("{0} is not a leap year".format(year))
```

21. Write a python program to multiply to matrices using nested loops.

```
# Program to multiply two matrices using nested loops

# 3x3 matrix
X = [[12,7,3],
     [4 ,5,6],
```



```
[7 ,8,9]]
# 3x4 matrix
Y = [[5,8,1,2],
      [6,7,3,0],
      [4,5,9,1]]
# result is 3x4
result = [[0,0,0,0],
          [0,0,0,0],
          [0,0,0,0]]

# iterate through rows of X
for i in range(len(X)):
    # iterate through columns of Y
    for j in range(len(Y[0])):
        # iterate through rows of Y
        for k in range(len(Y)):
            result[i][j] += X[i][k] * Y[k][j]

for r in result:
    print(r)
```



5.4 PREVIOUS UNIVERSITY QUESTION PAPERS

B **B3E049S**

Reg. No. _____ Name: _____

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY
THIRD SEMESTER B.TECH DEGREE EXAMINATION, JULY 2017

Course Code: **EE207**
Course Name: **COMPUTER PROGRAMMING (EE)**

Max. Marks: 100 Duration: 3 Hours

PART A
Answer all Questions.

1. a. Compare between compiler and assembler. (3)
b. Mention any four keywords and their meaning. (2)
2. Illustrate the syntax of while statement with an example. (5)
3. Explain how to initialize a 1-D numeric array and character array with examples. (5)
4. a. Enumerate three advantages of using functions. (3)
b. Give the purpose of return statement. (2)
5. Compare structure and union. (5)
6. What are pointers? Why they are used? Illustrate with an example. (5)
7. Give the syntax of fopen and fscanf to read data from a file. Illustrate with an example. (5)
8. Discuss on arithmetic operators in python. Give one example each. (5)

PART B
Answer any two questions.

9. a. Draw the flowchart and develop the algorithm for finding the area of a triangle by reading three sides (5)
b. Explain the different datatypes in C. (5)
10. a. Discuss the break and continue statement in C with an example. (5)
b. Write a C program to find the sum of digits of an integer, entered through the keyboard. (5)
11. a. Explain the syntax of switch statement with example. (5)
b. Write a C program to find the sum of all even numbers between two limits. (5)

PART C
Answer any two questions.

12. What are functions? Explains the different types of functions in detail with an example program for each type. (10)



SCMS SCHOOL OF ENGINEERING & TECHNOLOGY, KARUKUTTY

E

B3E049S

Pages: 2

13. Write a C program to find the transpose of a matrix. (10)
14. Explain the storage classes in C with appropriate example. (10)

PART D

Answer any two questions.

15. Compare structure and array and explain with an example. (10)
16. Write a C program to sort a set of mark sheets of 6 subjects. Make use of structure to develop the program and hence find the first three rank holders. (10)
17. a. Explain any five file handling functions and illustrate with an example. (5)
b. Explain the various data types in python. (5)



SCMS SCHOOL OF ENGINEERING & TECHNOLOGY, KARUKUTTY

B3E047

Pages: 2

Reg. No. _____ Name: _____

API ABDUL KALAM TECHNOLOGICAL UNIVERSITY
THIRD SEMESTER B.TECH DEGREE EXAMINATION, JANUARY 2017

Course Code: EE 207

Course Name: COMPUTER PROGRAMMING (EE)

Max. Marks: 100 Duration: 3 Hours

PART A

Answer all questions. 5 inerts enrA.

1. a) Discuss various data types used in C with examples. (3 marks)
- b) What do you understand by the term Keyword? (2 marks)
2. With suitable example discuss the use break and continue statements. (5 marks)
3. What do you mean by arrays? How they are initialized with declaration. (5 marks)
4. With suitable example explain what you understand by recursion. (5 marks)
5. Distinguish between structures and unions with an example. (5 marks)
6. Describe pointer variables. (5 marks)
7. Explain any 3 file handling operations in C programming. (5 marks)
8. Explain how variables are declared in python. (5 marks)

PART B

9. Differentiate Machine Language, Assembly Language and High level Language. (10 marks)
10. Write the algorithm and draw the flowchart to find the average height of boys and girls in a class from a given set of student data. (10 marks)
11. Discuss while, do while and for statement using suitable examples. (10 marks)



PART C

Solve the following questions. 10 marks each

12. Write a C program to sort the values of an array in descending order. (10 marks)
13. Write a C program to find the transpose of a matrix. (10 marks)
14. a) With proper examples explain the storage classes in C. (6 marks)
b) Differentiate the user defined and library functions. (4 marks)

PART D

Answer the following questions. 10 marks each

15. Write a C program to perform the file handling operation to read series integer number and write all odd numbers to a file to be called ODD and even numbers to EVEN. (10 marks)
16. Using functions write a program to swap the contents of two memory locations. (10 marks)
17. Write a python program to intake a simple calculator. (10 marks)

Sl. No.	Date & Day	Hour	Particulars of Portions Covered	Mode of Instruction	Signature
1.	3.01.2018	2,6	Introduction to C language	chalk & board	
2.	4.01.2018	5	Preprocessor directives, header files	chalk & board	
3.	5.01.2018	5	Datatypes and Qualifiers	chalk & board	
4.	8.01.2018	1	Operators, Constants & Variables	chalk & board	
5.	9.01.2018	3,7	Expressions - Arithmetic Expression, Type Conversions.	chalk & board	
6.	10.1.2018	2	Data input and Output - Printf & scanf	chalk & board	
7.	12.1.2018	5	Control Statements - if, if-else	chalk & board	
8.	15.1.2018	1	nested if, else if ladder	chalk & board	
9.	16.1.2018	1,2	Switch, goto, Conditional operator		
10.	17.1.2018	2	Looping - while, for	chalk & board	
11.	18.1.2018	3	do while - example program	chalk & board	
				Module 1	Completed
12.	19.1.2018	5	Arrays - initialization, Declaration	chalk & board	
13.	20.1.2018	3	Program to reverse an array, Merge two arrays - example program	chalk & board	
14.	22.01.2018	1,2	Two dimensional arrays - Matrix operations - addition	chalk & board	
15.	29.01.2018	1,7	Matrix Multiplication, diagonal sum, transpose of a matrix	chalk & board	

COURSE LOG

Sl. No.	Date & Day	Hour	Particulars of Portions Covered	Mode of Instruction	Signature
16	30.1.2018	3, 7	Strings - example pgm	chalk & board	<i>[Signature]</i>
17	31.1.2018	2	Table of, Sorting of Strings, Strings (enum), data type	chalk & board	<i>[Signature]</i>
18	1.2.2018	1	Structures - defining a structure, declaring a structure variable, Accessing structure elements	chalk & board	<i>[Signature]</i>
				Module 2	Completed
19	6.2.2018	3	Functions - Introduction	chalk & board	<i>[Signature]</i>
20	12.2.2018	1, 5	Function elements - Function definition, call & fn prototype	chalk & board	<i>[Signature]</i>
21	14.2.2018	1	Category of functions	chalk & board	<i>[Signature]</i>
22	16.2.2018	5	Function call by Value & reference	chalk & board	<i>[Signature]</i>
23	17.2.2018	6	Recursive function	chalk & board	<i>[Signature]</i>
24	19.2.2018	2	Example program for function	chalk & board	<i>[Signature]</i>
25	20.2.2018	3	Passing 1D and 2D arrays to functions	chalk & board	<i>[Signature]</i>
26	20.2.2018	7	Matrix addition & Sorting using fn.	chalk & board	<i>[Signature]</i>
27	21.2.2018	2	Pointers - Introduction	Module 4	Completed
28	22.2.2018	7	Declaration, Initialization & Accessing Value using pointer	chalk & board	<i>[Signature]</i>
29	23.2.2018	5	Example program - Swap d nos	chalk & board	<i>[Signature]</i>
30	24.2.2018	3	Chain of pointer, pointer Expression	chalk & board	<i>[Signature]</i>
31	26.2.2018	1	pointer increments & Scab jacob	chalk & board	<i>[Signature]</i>
32	27.2.2018	7	pointer and arrays (1D)	chalk & board	<i>[Signature]</i>

[Signature]
6/2/18

[Signature]
27/2/18

COURSE LOG

Sl. No.	Date & Day	Hour	Particulars of Portions Covered	Mode of Instruction	Signature
33	28.2.2018	2	Array of pointers, Structures of pointers	chalk & board	
34	1.3.2018	4	Example programs using pointers & structures	chalk & board	
				Module 3	Completed
35	7.3.2018	2	Sorting & searching - Bubble Sort	chalk & board	
36	9.3.2018	5	Selection Sort	chalk & board	
37	12.3.2018	1	Linear Search	chalk & board	
38	13.3.2018	3,7	Binary Search	"	
39	16.3.2018	5	Dynamic Mly Allocation	chalk & board	
40	17.3.2018	5	Scope rules - Storage class	chalk & board	
41	20.3.2018	3,7	static & Regular Variables	chalk & board	
				Module 5	Completed
42	21.3.2018	2	Data files - formatted, unformatted and text files	chalk & board	
43	26.3.2018	3	Text files - fopen, fclose	chalk & board	
44	27.3.2018	6,7	I/O functions in files - getc, put, getw, putw, fprintf, fscanf	chalk & board	
45	28.3.2018	4	Eg: Program - odd & even no	chalk & board	
46	3.4.2018	3	Random Access to files - fseek, rewind	chalk & board	
47	9.4.2018	1	Command Line arguments - eg	chalk & board	
48	10.4.2018	3	Previous Year Question paper		

Discussion:

Module 6 Completed

16/4/18

16/4/18

ATTENDANCE PARTICULARS

Date	Hour	Roll No. of Absentees
3.01.2018	2,6	5, 18, 21, 32, 41
4.01.2018	5	21, 27, 36, 41, 51
5.01.2018	5	21, 41
8.01.2018	1	8, 37
9.01.2018	3,7	8, 16, 33,
10.01.2018	2	7, 8, 17, 36, 47
12.01.2018	5	8, 10, 16, 20, 23, 36, 39, 42, 49
15.01.2018	1	7, 9, 15, 23, 49
16.01.2018	1,2	23, 47, 49
17.1.2018	2	Nil
18.1.2018	3	51
19.1.2018	5	6, 7, 24, 33, 37,
20.1.2018	3	3, 4, 9, 11, 26, 27, 33, 35, 36, 37, 40, 47, 51
22.1.2018	1,2	4, 7, 23, 37, 39, 48
29.1.2018	1,7	9, 12, 33, 37, 45
30.1.2018	3,7	29, 43,
31.1.2018	2	7, 17, 38,
1.2.2018	1	7, 17, 38
6.2.2018	3	1, 3, 4, 6, 10, 12, 15, 16, 17, 22, 32, 35, 37, 48, 49, 50
12.2.2018	1	2, 4, 7, 8, 11, 13, 16, 17, 25, 29, 31, 35, 36, 38, 45, 46, 48
12.2.2018	5	2, 4, 6, 7, 8, 11, 13, 14, 15, 16, 17, 25, 29, 31, 35, 36, 37, 38, 45, 46, 48,
14.2.2018	1	3, 11, 16
16.2.2018	5	14, 18, 24, 32, 36, 37, 49,
17.2.2018	6	3, 7, 14, 18, 24, 32, 37,
19.2.2018	2	Nil

ATTENDANCE PARTICULARS

Date	Hour	Roll No. of Absentees	Staff
20.2.2018	3	18,	2106-10-0
20.2.2018	7	18	2106-10-0
21.2.2018	2	28	2106-10-0
22.2.2018	7	32,37,46	2106-10-0
23.2.2018	5	17,	2106-10-0
24.2.2018	3	4,33,49	2106-10-0
26.2.2018	7	7,13	2106-10-0
27.2.2018	7	15	2106-10-0
28.2.2018	2	13,	2106-10-0
1.3.2018	4	1,3,4,8,10,12,15,16,17,19,21,22,24,29,32,33,36,37,41,45,50	
7.3.2018	2	35,37.	2106-10-0
9.3.2018	5	16,23,37,50	2106-10-0
12.3.2018	1	Nil	2106-10-0
13.3.2018	3	23,30,37	2106-10-0
13.3.2018	7	23,30,37	2106-10-0
16.3.2018	5	41,37	2106-10-0
17.3.2018	5	2,4,17,37,41,	2106-10-0
20.3.2018	3,7	37,42,	2106-10-0
21.3.2018	2	7,37,38,	2106-10-0
26.3.2018	3	1,7,11,35,37,	2106-6-01
27.3.2018	6,7	20,37,41,49	2106-6-01
28.3.2018	4	A,37	2106-6-01
3.4.2018	3	7,8,11,23,29,35,37,38,39,40,48,	2106-6-01
9.4.2018	1	1,4,7-10,12,15,17,20,22,32,36,37,42,43,45,50,	2106-6-01
10.4.2018	3	7,37,45	2106-6-01

PARTICULARS OF MARKS

Semester & Branch/Year of Admission: S₃ CSE

Course Name: Computer Programming

Class No.	Name	Assignment Marks			Internal Tests		Retest (if any)		Internal Academic Assessment Marks		
		Max 10	Max	Max	Max marks 20	Max marks 20	Max marks 20	Max marks 40	Assig nment Max 10	Internal Tests Max 40	Total Max 50
		1	2	3	1	2					
1	Maneesh Krishna M	10			11	20				22	32
2	Maroja E	10			17	20				37	47
3	Mariga Raphael	10			16	18				35	45
4	Mena C. Anil	10			19	20				30	40
5	Milna James	10			7	11				18	28
6	Mohammed Subaib VA	10			4	8	9			13	23
7	Muhammad Saheer cv	10			8	9				7	27
8	Neal A Vinod	10			9	14				24	34
9	Neethu Sunil	10			11	16				27	37
10	Nimisha Manoj	10			9	9				18	28
11	Nivedya k.p	10			7	15				22	32
12	Parvathi P.J	10			13	11				24	34
13	Pooja S	10			11	12				24	34
14	Premkathra Sujit	10			16	13				29	39
15	Rahul K.R	10			16	15				31	41
16	Ramachandran TH	10			6	3	7			13	23
17	Rithwik S Menon	10			8	10				18	28
18	Rizwana Yasmin Hashim	10			13	17				30	40
19	Robin Abraham	10			10	16				26	36
20	Rohan J. Thevara	10			17	14				32	42
21	Rohith Saju	10			10	9				19	29
22	Roumol Biju	10			11	16				27	37

PARTICULARS OF MARKS

Semester & Branch/Year of Admission: S₂ (SE) 2

Course Name: Computer Programming

Class No.	Name	Assignment Marks			Internal Tests		Retest (if any)		Internal Academic Assessment Marks		
		Max 10	Max	Max	Max marks 20	Max marks 20	Max marks 20	Max marks 40	Assig ment Max 10	Internal Tests Max 40	Total Max 50
		1	2	3	1	2					
23.	Roshan Prasad	10			16	18				35	45
24.	Saeed V. Bashier	10			15	17				32	42
25.	Sanju M.P	10			16	18				34	44
26.	Saranya M Nambiar	10			6	13				20	30
27.	Sarath A	10			14	16				30	40
28.	Sarath Arany Nair	10			14	12				26	36
29.	Sarath J	10			16	14				31	41
30.	Shalbt Mary T Ekha	10			14	14				28	38
31.	Shrifa Sekhar	10			15	16				31	41
32.	Shimil K Eldose	10			5	8				14	24
33.	Shravan Manoj	10			12	9				22	32
34.	Simon Saju	10			17	18				31	41
35.	Sneha S Nambiar	10			8	9				17	27
36.	Sohan James	10			4	4				13	23
37.	Sourav Binesh	10			0	1				1	11
38.	Sreeraj B.R	10			6	4				13	23
39.	Sten Benny	10			6	12				19	29
40.	Sumayya Subail	10			13	15				28	38
41.	Swathy Harish	10			14	9				23	33
42.	Thomas Vilangadan	10			16	15				31	41
43.	Varun S Nair	10			14	17				31	41
44.	Vikhresh Krishna	10			10	14				24	34

SCMS SCHOOL OF ENGINEERING & TECHNOLOGY, KARUKUTTY
S2 Computer Science and Engineering - II (B17)
Consolidated Attendance

From : 01/01/2018 to 12/04/2018

CLN e.	NAME	Course Code	MA102	PH100	BE110	BE102	CS100	CE100	PH110	CS120	CE 110
		Roll No	Differential Equations	Engineering Physics	Engineering Graphics	Design and Engineering	Computer Programming	Basics of Civil Engineering	Engineering Physics Lab	Computer Programming Lab	Civil Engineering Workshop
1	MANEESH KRISHNA M	SCS/7789/17	94	95	92	93	94	90	100	100	100
2	MANOJA E	SCS/7928/17	92	96	96	98	96	95	100	100	100
3	MARIYA RAPHEL	SCS/7771/17	94	95	93	94	94	88	100	93	100
4	MEERA C ANIL	SCS/7616/17	82	88	80	91	84	77	100	98	100
5	MILNA JAMES	SCS/8051/17	95	99	99	100	97	97	100	100	100
6	MOHAMMED SUHAIB V A	SCS/7643/17	95	97	92	98	96	90	82	100	100
7	MUHAMMAD SAHEER C V	SCS/8057/17	76	83	77	80	79	82	100	85	86
8	NEAL A.VINOD	SCS/7637/17	79	85	81	81	84	80	100	78	86
9	NEETHU SUNIL	SCS/8011/17	94	96	95	85	93	92	100	100	100
10	NIMISHA MANOJ	SCS/7609/17	94	92	91	93	93	88	100	90	86
11	NIVEDYA K P	SCS/7748/17	87	92	95	91	93	87	100	100	86
12	PARVATHI P J	SCS/7977/17	92	93	89	89	93	87	100	100	100
13	POOJA S	SCS/7613/17	85	91	92	96	93	90	100	93	100
14	PREMKRISHNA SUJIT	SCS/7597/17	97	100	98	98	97	97	100	100	86
15	RAHUL K R	SCS/7784/17	92	92	91	91	93	88	100	100	100
16	RAMACHANDRAN T H	SCS/7618/17	85	83	81	85	87	78	82	93	86
17	RITHWIK S MENON	SCS/7630/17	89	87	87	91	86	85	100	90	76
18	RIZWANA YASMIN HASHIM	SCS/7723/17	94	97	97	100	94	95	100	100	100
19	ROBIN ABRAHAM	SCS/7632/17	90	96	89	94	99	90	100	100	100
20	ROHAN J THEVARA	SCS/7801/17	89	87	96	87	94	90	100	85	100
21	ROHITH SAJU	SCS/7631/17	87	89	93	94	93	93	82	93	100
22	ROSEMOL BIJU	SCS/7830/17	95	92	91	94	96	90	100	100	100
23	ROSHAN PRASAD	SCS/8006/17	89	83	83	81	84	83	82	93	100
24	SAEED V BASHEER	SCS/7584/17	97	99	98	98	97	98	100	100	100
25	SANJU M P	SCS/7582/17	95	97	98	96	97	95	100	100	100
26	SARANYA M. NAMBIAR	SCS/7606/17	95	99	95	94	99	95	100	100	100
27	SARATH A	SCS/8022/17	94	96	93	94	97	92	100	100	100
28	SARATH AMAY NAIR	SCS/7604/17	97	100	100	98	99	97	100	100	100
29	SARATH J	SCS/7768/17	82	91	82	85	91	80	100	92	100
30	SHALLET MARY T ELDHO	SCS/7773/17	98	96	98	100	97	97	100	100	100
31	SHILPA SEKHAR	SCS/7732/17	95	97	97	98	97	95	100	100	100
32	SHIMIL K ELDOSE	SCS/7635/17	90	91	89	93	87	87	75	81	100
33	SHRAVAN MANOJ	SCS/7591/17	90	92	95	93	91	87	100	100	86

CLN No	NAME	Course Code	MA102	PH100	BE110	BE102	CS100	CE100	PH110	CS120	CE 110
		Roll No	Differential Equations	Engineering Physics	Engineering Graphics	Design and Engineering	Computer Programming	Basics of Civil Engineering	Engineering Physics Lab	Computer Programming Lab	Civil Engineering Workshop
34	SIMON SAJU	SCS/7583/17	97	100	96	98	100	95	100	100	100
35	SNEHA S NAMBIAR	SCS/7996/17	87	91	92	89	90	85	100	92	100
36	SOHAN JAMES	SCS/7581/17	84	85	88	83	86	85	75	81	86
37	SOURAV BINESH	SCS/7628/17	56	58	47	59	54	57	57	43	71
38	SREERAJ B R	SCS/8025/17	90	92	91	94	93	88	86	89	100
39	STEN BENNY	SCS/8003/17	90	95	93	93	94	92	86	100	100
40	SUMAYYA SUHAIL	SCS/7589/17	94	97	97	96	97	95	100	100	100
41	SWATHY HARISH	SCS/7588/17	81	86	91	93	87	92	86	97	86
42	THOMAS VILANGADAN	SCS/7619/17	94	92	94	91	91	93	86	97	100
43	VARUN S NAIR	SCS/8026/17	92	93	92	94	94	92	86	100	100
44	VIKHESH KRISHNA	SCS/7590/17	94	96	92	94	99	90	100	100	100
45	VINAY ARUN DEO	SCS/7585/17	89	95	89	91	90	90	100	92	100
46	VINAY STEEPHEN	SCS/8035/17	87	89	87	89	97	88	86	84	100
47	VINIL VARGHESE	SCS/8021/17	90	97	93	91	94	87	100	92	100
48	VISHNU GOPIDAS	SCS/7744/17	92	97	91	94	91	90	100	100	100
49	VISHNUPRASAD R	SCS/8043/17	92	89	91	87	86	90	86	92	100
50	VISHNURAJ K R	SCS/7767/17	95	95	87	96	94	90	100	100	100
51	VISHNU T P	SCS/7595/17	89	93	88	91	96	87	100	100	100


FACULTY


12/4/18


PRINCIPAL
12/4

CS/F07.0

SCMS SCHOOL OF ENGINEERING & TECHNOLOGY, KARUKUTTY

B.Tech - Computer Science and Engineering. KTU 2018 S2 - B

CS100 COMPUTER PROGRAMMING

SESSIONAL MARK LIST-Theory

Roll No	Name	Admission No	Assignment 10 Marks	Test 40 Marks	Total 50Marks
1	MANEESH KRISHNA M	SCS/7789/17	10	22	32
2	MANOJA E	SCS/7928/17	10	37	47
3	MARIYA RAPHEL	SCS/7771/17	10	35	45
4	MEERA C ANIL	SCS/7616/17	10	30	40
5	MILNA JAMES	SCS/8051/17	10	18	28
6	MOHAMMED SUHAIB V A	SCS/7643/17	10	13	23
7	MUHAMMAD SAHEER C V	SCS/8057/17	10	17	27
8	NEAL A.VINOD	SCS/7637/17	10	24	34
9	NEETHU SUNIL	SCS/8011/17	10	27	37
10	NIMISHA MANOJ	SCS/7609/17	10	18	28
11	NIVEDYA K P	SCS/7748/17	10	22	32
12	PARVATHI P J	SCS/7977/17	10	24	34
13	POOJA S	SCS/7613/17	10	24	34
14	PREMKRISHNA SUJIT	SCS/7597/17	10	29	39
15	RAHUL K R	SCS/7784/17	10	31	41
16	RAMACHANDRAN T H	SCS/7618/17	10	13	23
17	RITHWIK S MENON	SCS/7630/17	10	18	28
18	RIZWANA YASMIN HASHIM	SCS/7723/17	10	30	40
19	ROBIN ABRAHAM	SCS/7632/17	10	26	36
20	ROHAN J THEVARA	SCS/7801/17	10	32	42
21	ROHITH SAJU	SCS/7631/17	10	19	29
22	ROSEMOL BIJU	SCS/7830/17	10	27	37
23	ROSHAN PRASAD	SCS/8006/17	10	35	45
24	SAEED V BASHEER	SCS/7584/17	10	32	42
25	SANJU M P	SCS/7582/17	10	34	44
26	SARANYA M. NAMBIAR	SCS/7606/17	10	20	30
27	SARATH A	SCS/8022/17	10	30	40
28	SARATH AMAY NAIR	SCS/7604/17	10	26	36
29	SARATH J	SCS/7768/17	10	31	41
30	SHALLET MARY T ELDHO	SCS/7773/17	10	28	38
31	SHILPA SEKHAR	SCS/7732/17	10	31	41
32	SHIMIL K ELDOSE	SCS/7635/17	10	14	24
33	SHRAVAN MANOJ	SCS/7591/17	10	22	32
34	SIMON SAJU	SCS/7583/17	10	31	41

35	SNEHA S NAMBIAR	SCS/7996/17	10	17	27
36	SOHAN JAMES	SCS/7581/17	10	13	23
37	SOURAV BINESH	SCS/7628/17	10	1	11
38	SREERAJ B R	SCS/8025/17	10	13	23
39	STEN BENNY	SCS/8003/17	10	19	29
40	SUMAYYA SUHAIL	SCS/7589/17	10	28	38
41	SWATHY HARISH	SCS/7588/17	10	23	33
42	THOMAS VILANGADAN	SCS/7619/17	10	31	41
43	VARUN S NAIR	SCS/8026/17	10	31	41
44	VIKHNEESH KRISHNA	SCS/7590/17	10	24	34
45	VINAY ARUN DEO	SCS/7585/17	10	22	32
46	VINAY STEEPHEN	SCS/8035/17	10	17	27
47	VINIL VARGHESE	SCS/8021/17	10	22	32
48	VISHNU GOPIDAS	SCS/7744/17	10	17	27
49	VISHNUPRASAD R	SCS/8043/17	10	28	38
50	VISHNURAJ K R	SCS/7767/17	10	27	37
51	VISHNU T P	SCS/7595/17	10	8	18


FACULTY IN CHARGE


HOD


PRINCIPAL

REMEDIAL MEASURES TAKEN

NAME OF SUBJECT:

Computer Programming

Batch No:

17

Topics : Programs using looping, decision making

Date : 28-2-18

Time : 4.30-6.30

Roll No	Name	Sign
5	MILNA JAMES	<i>Milna James</i>
6	MOHAMMED SUHAIB V A	<i>Mohammed Suhaib</i>
7	MUHAMMAD SAHEER C V	<i>Saheer</i>
16	RAMACHANDRAN T H	<i>Ramachandran</i>
17	RITHWIK S MENON	<i>Rithwik</i>
35	SNEHA S NAMBIAR	<i>Sneha</i>
36	SOHAN JAMES	<i>Sohan</i>
51	VISHNU T P	<i>Vishnu</i>

BINI OMMAN *Bini*

REMEDIAL MEASURES TAKEN

NAME OF SUBJECT: Computer Programming

Batch No: 17

Topics : Arrays-1D,2D

Date : 16-4-18

Time : 4.30-6.30

Roll No	Name	Sign
5	MILNA JAMES	<i>Milna James</i>
6	MOHAMMED SUHAIB V A	<i>Mohammed Suhaib V A</i>
7	MUHAMMAD SAHEER C V	<i>Muhammad Saheer C V</i>
16	RAMACHANDRAN T H	<i>Ramachandran T H</i>
17	RITHWIK S MENON	<i>Rithwik S Menon</i>
35	SNEHA S NAMBIAR	<i>Sneha S Nambiar</i>
36	SOHAN JAMES	<i>Sohan James</i>
51	VISHNU T P	<i>Vishnu T P</i>



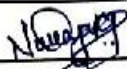
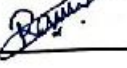




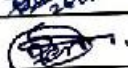
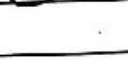


BINI OMMAN *Bini Ommann*

REMEDIAL CLASS-PHASE 1

Class:S2CS2

Subject: Computer Programming

Date :26/02/2018

Sl.no	Name	Signature
1	MILNA JAMES	
2	MOHAMMED SUHAIB V A	
3	MUHAMMAD SAHEER C V	
4	NIVEDYA K P	
5	RAMACHANDRAN T H	
6	RITHWIK S MENON	
7	SARANYA M. NAMBIAR	
8	SHIMIL K ELDOSE	
9	SNEHA S NAMBIAR	
10	SOHAN JAMES	
11	SOURAV BINESH	
12	SREERAJ B R	
13	STEN BENNY	
14	VINAY ARUN DEO	
15	VINAY STEEPHEN	
16	VISHNU GOPIDAS	
17	VISHNU T P	

- 18. Rosemol Biju
- 19. Neethu Sunil
- 20. Sunayya Sachil
- 21. Maneesh Krishna.








APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY

Academic Calendar July 2017- July 2018

(B.Tech, B.Arch, M.Tech, M.Arch, M.Planning, MCA and Evening B.Tech&M.Tech)
Please see separate Academic Calendar for MBA

Day	April 2018	May 2018	June 2018	July 2018
Mon				
Tue				
Wed		1 May Day		
Thu		2 Exam S1 / S6 Slot D		
Fri		3 Exam S4 Slot C		
Sat		4 Exam S1 Slot E1/S6 Slot F	1 Commencement of Supplementary Exams	
Sun		5	2 Exam S1 Slot A	
Mon	1 Easter	6	3	
Tue	2	7	4 Exam S4 Slot D	1
Wed	3	8 Exam S2 Slot E2/S6 Slot F	5 Exam S3 Slot A	2 Exam S6 Slot A
Thu	4	9 Exam S4 Slot E	6 M.Tech/M.Arch/M.Planning Viva begins	3 Exam S6 Slot B
Fri	5	10 Exam S2 Slot E3 - Commencement of Summer Courses	7 Exam S3 Slot B	4 Exam S6 Slot C
Sat	6 Course Committee/Class Committee Meeting	11 Exam S2 Slot F1- Last date for submission of project report in the college (M.Tech/M.Arch/M.Planning)	8 Exam S1 Slot E	5 Exam S6 Slot D
Sun	7 College level Arts fest To be completed	12	9 Exam S3 Slot C	6 Exam S6 Slot E
Mon	8 Publish Internal Marks, Summer Course Registration	13	10 B.Tech S4 result declaration	
Tue	9 Last date for evaluation of Jury/Practical's	14	11	7
Wed	10 Classes End, Publish Attendance	15 Exam S2 Slot F2	12 Exam S3 Slot D	8 Exam S6 Slot F
Thu	11 Forward Attendance & Internal Marks to KTU	16 University Arts Fest	13 Exam S3 Slot E	10 Exam S2 Slot A
Fri	12 Vishu	17	14 B.Tech S5 result declaration	12 Exam S1/S2 Slot B1
Sat	13	18	15 Id-ul-Fitr	13 Exam S1/S2 Slot B2
Sun	14	19	16	14
Mon	15	20	17	15
Tue	16	21 Exam S5 Slot A (Suppl'y)	18	16 Exam S1/S2 Slot C1
Wed	17	22 Exam S5 Slot B (Suppl'y)	19	17 Exam S1/S2 Slot C2
Thu	18	23 Exam S5 Slot C (Suppl'y)	20 M.Tech/M.Arch/M.Planning Viva ends	18 Exam S2 Slot D
Fri	19	24 Exam S5 Slot D (Suppl'y)	21 Exam S3/S4 Slot F1	19 Exam S1/S2 Slot E1
Sat	20	25 Exam S5 Slot E (Suppl'y)	22 B.Tech S2 Result Declaration	20 Exam S1/S2 Slot E2
Sun	21	26	23	21
Mon	22	27	24	22
Tue	23 Commencement of S2/S4/S6 Exams	28 Exam S5 Slot F (Suppl'y)	25 Exam S4 Slot A	23 Exam S2 Slot E3
Wed	24 Exam S4 Slot F	29 Last date for M.Tech/M.Arch/M.Planning Project report to the university by the principal	26 Exam S4 Slot B	24 Exam S1/S2 Slot F1
Thu	25 Exam S2 / S6 Slot B	30 Summer Courses Ends	27 Exam S4 Slot C	25 Exam S1/S2 Slot F2
Fri	26 Exam S4 slot A Last date for M.Tech/M.Arch/M.Planning Project evaluation in the department committee	31 Report Eligibility of Students after Summer Course	28 Exam S4 Slot D	
Sat	27 Exam S2/ S6 Slot C		29 Publication of M.Tech/M.Arch/M.Planning Results	27
Sun	28		30 Exam S4 Slot E	28
Mon	29			29
Tue	30 Exam S4 Slot B			30
Wed				31



APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY

Academic Calendar July 2017- July 2018

(B.Tech, B.Arch, M.Tech, M.Arch, M.Planning, MCA and Evening B.Tech&M.Tech)
Please see separate Academic Calendar for MBA

Day	January 2018	February 2018	March 2018
Mon	1 Registration Starts Commencement of Even Semester Classes		
Tue	2 Munnam Jayanthi		
Wed	3		
Thu	4	1	1
Fri	5	2 Publish Attendance	2 Publish Attendance
Sat	6	3	3
Sun	7	4	4
Mon	8 Course Committee/Class Committee Meeting	5	5
Tue	9	6	6
Wed	10	7	7
Thu	11	8	8 Last date for forwarding the list of the external examiner to the University by the cluster conveners (M.Tech/M.Arch/M.Planning)
Fri	12 Registration Ends	9 B.Tech S1/S3/S5 result declaration	9
Sat	13	10 Test 1 to be Completed	10 Test 2 to be Completed
Sun	14	11	11
Mon	15	12	12
Tue	16	13 Maha Shivratri	13
Wed	17	14 Publish Test 1 Marks	14
Thu	18	15	15
Fri	19	16 Tech Fest	16 Publish Test 2 Marks
Sat	20	17	17
Sun	21	18	18
Mon	22	19	19
Tue	23	20	20
Wed	24	21	21
Thu	25	22	22
Fri	26 Republic Day	23	23
Sat	27	24	24
Sun	28	25	25
Mon	29	26	26
Tue	30	27	27
Wed	31	28	28
Thu			29 Maundy Thursday
Fri			30 Good Friday
Sat			31
Sun			
Mon			
Tue			

SCMS SCHOOL OF ENGINEERING & TECHNOLOGY, KARUKUTTY

S2 Computer Science and Engineering (B17)

Checklist: Course File

CS100 COMPUTER PROGRAMMING

SL NO	CONTENTS	STATUS
1	COURSE DIARY	✓
	1.1 FACULTY TIMETABLE	✓
	1.2 DETAILED COURSE PLAN	✓
	1.3 SYLLABUS	✓
	1.4 INTERNAL TEST1 MARKS	✓
	1.5 INTERNAL TEST 2 MARKS	✓
	1.6 ACADEMIC CALENDAR	✓
	1.7 FINAL INTERNAL ACADEMIC ASSESSMENT	✓
	1.8 FINAL SUBJECT ATTENDANCE	✓
2	INTERNAL TEST 1 QUESTION PAPER	✓
3	INTERNAL TEST 2 QUESTION PAPER	✓
4	INTERNAL TEST 1 ANSWER KEY	✓
5	INTERNAL TEST 2 ANSWER KEY	✓
6	ASSIGNMENT QUESTION	✓
7	INDUSTRIAL RELEVANCE	✓
8	STUDENT ASSIGNMENT SHEETS	✓
9	STUDENT ANSWER SHEETS	✓
10	COURSE NOTES	✓



**SCMS SCHOOL OF ENGINEERING AND TECHNOLOGY
VIDYA NAGAR, KARUKUTTY, ERNAKULAM
DEPARTEMENT OF COMPUTER SCIENCE AND ENGINEERING
CO ASSESSMENT TOOL**

Course Code & Name : :CS100 COMPUTER PROGRAMMING
Faculty : BINI OMMAN.
Academic Year : 2017-18
Class : S2-CSE-II
Regulation : R-2015

COURSE OUTCOMES -:CS100 COMPUTER PROGRAMMING

After the completion of this course, students should be able to

1. Identify appropriate C language constructs to solve problems
2. Thorough understanding of arrays and strings
3. Explain the concept of Pointers and their applications
4. Analyze problems, identify subtasks and implement them as functions/procedures
5. Apply sorting and searching techniques to solve application programs
6. Explain the concept of file system for handling data storage and apply it for solving problems

Mapping of Assessment tools with COs

Assessment Tools	CO1	CO2	CO3	CO4	CO5	CO6
IAT	X	X		X	X	
Class Test			X			X
Assignment	X		X	X		
Concept Test						
Rubrics						X

Weightage for Assessment tools

Course Outcomes	IAT	Concept Test	Assignment	Rubrics	Class Test
CS100.1	80		20		
CS100.2	100				
CS100.3			20		80
CS100.4	80		20		
CS100.5	100				
CS100.6				20	80



**SCMS SCHOOL OF ENGINEERING AND TECHNOLOGY.KARUKUTTY
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

CS100 : COMPUTER PROGRAMMING - BATCH 17 CS2

Question	Responses					Sum 5+4
	5.To a very great extent	4.To a great extent	3.To a moderate extent	2.To some extent	1.Not at all	
CS100.1 I am able to identify appropriate C language constructs to solve problems	17	20	11	0	1	75.51
	34.69	40.82	22.45	0.00	2.04	
CS100.2 I am able to understand the concept of arrays and strings	17	19	12	1	0	73.47
	34.69	38.78	24.49	2.04	0.00	
CS100.3 I am able to understand concept of Pointers and their applications	15	20	14	0	0	71.43
	30.61	40.82	28.57	0.00	0.00	
CS100.4 I am able to analyze problems, identify subtasks and implement them as functions/ procedures	17	25	6	0	1	85.71
	34.69	51.02	12.24	0.00	2.04	
CS100.5 I am able to apply sorting and searching techniques to solve application programs	16	20	12	0	1	73.47
	32.65	40.82	24.49	0.00	2.04	
CS100.6 I am able to explain the concept of file system for handling data storage and apply it for solving problems	17	19	12	0	1	73.47
	34.69	38.78	24.49	0.00	2.04	

**SCMS SCHOOL OF ENGINEERING & TECHNOLOGY, KARUKUTTY
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
ASSESSMENT OF ATTAINMENT OF COURSE OUTCOMES**

Course Code & Name :CS100 COMPUTER PROGRAMMING
Faculty :BINI OMMAN
Academic Year :2018-2019
Class :S2CS
Regulation : 2016

After the completion of this course, students should be able to

S.NO	REG. NO	NAME	Course Outcomes																		University exam					
			CS100.1			CS100.2			CS100.3			CS100.4			CS100.5			CS100.6								
			80	20	100	100.0	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100			
			IAT - I	ASSIGNMEN T I	TOTAL	IAT - II	TOTAL	TOTAL	ASSIGNMEN T II	CT - I	TOTAL	IAT - II	ASSIGNMEN T II	CT - I	TOTAL	IAT - II	ASSIGNMEN T II	CT - I	TOTAL	IAT - II	ASSIGNMEN T II	CT - 2	TOTAL	RUBRICS	TOTAL	University Grade
1	SCM17CS056	MANEESH KRISHNA M	44.0	20.0	64.0	46.4	46.4	46.4	46.4	20	72.0	92.0	51.1	20	71.1	37.5	37.5	64	37.5	37.5	64	16.0	80	80	80	B
2	SCM17CS057	MANOJA E	68.8	20.0	88.8	96.4	96.4	96.4	20	80.0	100.0	80.0	20	100.0	87.5	87.5	87.5	80	87.5	87.5	80	20.0	100	100	100	O
3	SCM17CS058	MARIYA RAPHEL	65.6	20.0	85.6	100.0	100.0	100.0	20	80.0	100.0	66.7	20	86.7	75.0	75.0	75.0	80	75.0	75.0	80	20.0	100	100	100	O
4	SCM17CS059	MEERA C ANIL	76.8	20.0	96.8	64.3	64.3	64.3	20	80.0	100.0	35.6	20	55.6	0.0	0.0	0.0	72	0.0	0.0	72	18.0	90	90	90	B
5	SCM17CS060	MILNA JAMES	28.0	20.0	48.0	46.4	46.4	46.4	20	64.0	84.0	55.6	20	75.6	37.5	37.5	37.5	56	37.5	37.5	56	14.0	70	70	70	C
6	SCM17CS061	MOHAMMED SUHAIB V A	16.0	20.0	36.0	37.5	37.5	37.5	20	48.0	68.0	35.6	20	55.6	0.0	0.0	0.0	40	0.0	0.0	40	10.0	50	50	50	F
7	SCM17CS062	MUHAMMAD SAHEER C V	32.0	20.0	52.0	46.4	46.4	46.4	20	56.0	76.0	31.1	20	51.1	62.5	62.5	62.5	48	62.5	62.5	48	12.0	60	60	60	C

8	SCM17CS063	NEAL A VINOD	36.8	20.0	56.8	78.6	78.6	20	72.0	92.0	57.8	20	77.8	25.0	25.0	64	16.0	80	C
9	SCM17CS064	NEETHU SUNIL	43.2	20.0	63.2	78.6	78.6	20	80.0	100.0	66.7	20	86.7	50.0	50.0	72	18.0	90	B
10	SCM17CS065	NIMISHA MANOJ	36.0	20.0	56.0	41.1	41.1	20	64.0	84.0	44.4	20	64.4	25.0	25.0	56	14.0	70	C
11	SCM17CS066	NIVEDYA K P	26.4	20.0	46.4	82.1	82.1	20	72.0	92.0	62.2	20	82.2	25.0	25.0	64	16.0	80	C
12	SCM17CS067	PARVATHI P J	52.8	20.0	72.8	44.6	44.6	20	80.0	100.0	60.0	20	80.0	25.0	25.0	72	18.0	90	B
13	SCM17CS068	POOJA S	44.8	20.0	64.8	48.2	48.2	20	72.0	92.0	66.7	20	86.7	50.0	50.0	64	16.0	80	B
14	SCM17CS069	PREMKRISHNA SUJIT	62.4	20.0	82.4	71.4	71.4	20	64.0	84.0	42.2	20	62.2	50.0	50.0	56	14.0	70	C
15	SCM17CS070	RAHUL K R	64.0	20.0	84.0	67.9	67.9	20	80.0	100.0	75.6	20	95.6	25.0	25.0	72	18.0	90	B+
16	SCM17CS071	RAMACHANDRAN T H	21.6	20.0	41.6	5.4	5.4	20	40.0	60.0	26.7	20	46.7	12.5	12.5	32	8.0	40	F
17	SCM17CS072	RITHVIK S MENON	31.2	20.0	51.2	46.4	46.4	20	48.0	68.0	53.3	20	73.3	12.5	12.5	40	10.0	50	P
18	SCM17CS073	RIZWANA YASMIN HASHIM	52.0	20.0	72.0	82.1	82.1	20	72.0	92.0	73.3	20	93.3	75.0	75.0	64	16.0	80	B+
19	SCM17CS074	ROBIN ABRAHAM	37.6	20.0	57.6	83.9	83.9	20	64.0	84.0	71.1	20	91.1	12.5	12.5	56	14.0	70	C
20	SCM17CS075	ROHAN J THEVARA	68.8	20.0	88.8	57.1	57.1	20	72.0	92.0	75.6	20	95.6	75.0	75.0	64	16.0	80	B+
21	SCM17CS076	ROHITH SAJU	40.8	20.0	60.8	55.4	55.4	20	56.0	76.0	31.1	20	51.1	0.0	0.0	48	12.0	60	c
22	SCM17CS077	ROSEMOL BIJU	44.8	20.0	64.8	87.5	87.5	20	80.0	100.0	62.2	20	82.2	12.5	12.5	72	18.0	90	B
23	SCM17CS078	ROSHAN PRASAD	65.6	20.0	85.6	87.5	87.5	20	80.0	100.0	75.6	20	95.6	100.0	100.0	72	18.0	90	B+
24	SCM17CS079	SAEED V BASHEER	60.8	20.0	80.8	85.7	85.7	20	80.0	100.0	62.2	20	82.2	100.0	100.0	72	18.0	90	B+
25	SCM17CS080	SANJU M P	62.4	20.0	82.4	82.1	82.1	20	80.0	100.0	80.0	20	100.0	87.5	87.5	72	18.0	90	B
26	SCM17CS081	SARANYA M NAMBIAR	25.6	20.0	45.6	67.9	67.9	20	72.0	92.0	48.9	20	68.9	62.5	62.5	64	16.0	80	B
27	SCM17CS082	SARATH A	57.6	20.0	77.6	83.9	83.9	20	80.0	100.0	53.3	20	73.3	100.0	100.0	72	18.0	90	B+
28	SCM17CS083	SARATH AMAY NAIR	55.2	20.0	75.2	62.5	62.5	20	80.0	100.0	44.4	20	64.4	62.5	62.5	72	18.0	90	B
29	SCM17CS084	SARATH J	65.6	20.0	85.6	82.1	82.1	20	72.0	92.0	55.6	20	75.6	12.5	12.5	64	16.0	80	B+
30	SCM17CS085	SHALLET MARY T ELDHO	56.0	20.0	76.0	53.6	53.6	20	80.0	100.0	75.6	20	95.6	100.0	100.0	80	20.0	100	B+
31	SCM17CS086	SHILPA SEK HAR	59.2	20.0	79.2	82.1	82.1	20	80.0	100.0	62.2	20	82.2	75.0	75.0	72	18.0	90	B
32	SCM17CS087	SHIMIL K ELDJOSE	20.0	20.0	40.0	60.7	60.7	20	64.0	84.0	15.6	20	35.6	0.0	0.0	56	14.0	70	C
33	SCM17CS088	SHRAVAN MANOJ	48.8	20.0	68.8	51.8	51.8	20	72.0	92.0	26.7	20	46.7	50.0	50.0	64	16.0	80	B+
34	SCM17CS089	SIMON SAJU	50.4	20.0	70.4	94.6	94.6	20	80.0	100.0	75.6	20	95.6	12.5	12.5	72	18.0	90	B+

35	SCM17CS090	SNEHA S NAMBIAR	31.2	20.0	51.2	30.4	30.4	20	48.0	68.0	46.7	20	66.7	87.5	87.5	40	10.0	50	C
36	SCM17CS091	SOHAN JAMES	15.2	20.0	35.2	16.1	16.1	20	48.0	68.0	26.7	20	46.7	0.0	0.0	40	10.0	50	F
38	SCM17CS093	SREERAJ B R	21.6	20.0	41.6	35.7	35.7	20	64.0	84.0	4.4	20	24.4	0.0	0.0	56	14.0	70	P
39	SCM17CS094	STEN BENNY	24.8	20.0	44.8	67.9	67.9	20	64.0	84.0	48.9	20	68.9	25.0	25.0	56	14.0	70	C
40	SCM17CS095	SUMAYYA SUHAIL	50.4	20.0	70.4	80.4	80.4	20	80.0	100.0	62.2	20	82.2	0.0	0.0	72	18.0	90	B+
41	SCM17CS096	SWATHY HARISH	55.2	20.0	75.2	35.7	35.7	20	80.0	100.0	53.3	20	73.3	12.5	12.5	72	18.0	90	B
42	SCM17CS097	THOMAS VILANGADAN	63.2	20.0	83.2	71.4	71.4	20	72.0	92.0	80.0	20	100.0	0.0	0.0	64	16.0	80	B
43	SCM17CS098	VARUN S NAIR	54.4	20.0	74.4	87.5	87.5	20	80.0	100.0	62.2	20	82.2	100.0	100.0	72	18.0	90	A
44	SCM17CS099	VIKHNESH KRISHNA	40.0	20.0	60.0	85.7	85.7	20	64.0	84.0	44.4	20	64.4	0.0	0.0	56	14.0	70	C
45	SCM17CS100	VINAY ARUN DEO	32.0	20.0	52.0	82.1	82.1	20	64.0	84.0	46.7	20	66.7	12.5	12.5	56	14.0	70	B
46	SCM17CS101	VINAY STEEPHEN	21.6	20.0	41.6	64.3	64.3	20	64.0	84.0	42.2	20	62.2	12.5	12.5	56	14.0	70	P
47	SCM17CS102	VINIL VARGHESE	40.0	20.0	60.0	75.0	75.0	20	72.0	92.0	44.4	20	64.4	0.0	0.0	64	16.0	80	C
48	SCM17CS103	VISHNU GOPIDAS	23.2	20.0	43.2	66.1	66.1	20	64.0	84.0	40.0	20	60.0	12.5	12.5	56	14.0	70	P
49	SCM17CS104	VISHNU PRASAD R	57.6	20.0	77.6	67.9	67.9	20	72.0	92.0	46.7	20	66.7	100.0	100.0	64	16.0	80	B
50	SCM17CS105	VISHNU RAJ K R	44.8	20.0	64.8	78.6	78.6	20	80.0	100.0	68.9	20	88.9	25.0	25.0	72	18.0	90	B
51	SCM17CS106	VISHNU T P	0.0	20.0	20.0	0.0	0.0	20	40.0	60.0	31.1	20	51.1	0.0	0.0	32	8.0	40	F
No of Students scored set attainment level			23	50	31	32.0	32	50	48	50	30	50	40	16	16	44	44	44	29
% of Students scored set attainment level			46	100	62	64	64	100	96	100	60	100	80	32	32	88	88	88	58
Level of Attainment			0	3	1	1	1	3	3	3	1	3	3	0	0	3	3	3	0
Course			Level 1: 60 % of Students scored more than set attainment level																
Outcomes			Level 2: 70 % of Students scored more than set attainment level																
CS110.1			Level 3: 80 % of Students scored more than set attainment level																
CS110.2																			
CS110.3																			
CS110.4																			
CS110.5																			
CS110.6																			
			1.60																

Level 1: 60 % of Students scored more than set attainment level	
Level 2: 70 % of Students scored more than set attainment level	
Level 3: 80 % of Students scored more than set attainment level	

WEIGHTAGE	
INTERNAL	33%
UNIVERSITY	67%

Note:

the same course for KTU since 2015, or the no. of different classes for which

3. Only those for which University results are out need to be included.

Reg. No. _____ Name: _____

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY
SECOND SEMESTER B.TECH DEGREE EXAMINATION, MAY 2017

Course Code: CS100

Course Name: COMPUTER PROGRAMMING (CS, IT)

Max. Marks: 100

Duration: 3 Hours

PART A*Answer all Questions.*

1. What are identifiers? Give the rules for forming identifiers in C. (3)
2. What do you mean by associativity? What is the associativity of unary operators?(2)
3. Explain the use of continue statement with the help of an example. (3)
4. What are library functions? (2)
5. What is the use of a function prototype? Give the function prototype of a function accepting one float value and an integer array and return a float array. (3)
6. Write a function to compute the length of a string declared using a pointer variable.(3)
7. How do you initialize a two dimensional array during declaration? (2)
8. What are enumerated data types? How ordinal values are assigned to its members?(3)
9. Explain the difference between *ptr++ and (*ptr)++ if ptr is pointing to the first element of an integer array. (3)
10. With an example, show how you can access the members of a structure variable using a pointer to the variable. (2)
11. Explain any three file opening modes? (2)
12. How can you perform read and write operations on an unformatted data file? (3)
13. Explain static storage class with the help of an example. (2)
14. Explain the arguments passed to the main function as command line arguments. (3)
15. What is a NULL pointer? (2)
16. Explain the use of indirection operator with the help of an example. (2)

Total Pages: 2

Reg No.: _____

Name: _____

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY
SECOND SEMESTER B.TECH DEGREE EXAMINATION, JULY 2017

Course Code: CS100**Course Name: COMPUTER PROGRAMMING (CS, IT)**

Max. Marks: 100

Duration: 3 Hours

PART A*Answer all Questions.*

- 1 What is a variable? How are the variables declared in C? (2)
- 2 How does x++ differ from ++x? Explain with suitable examples. (3)
- 3 Give the declaration of variable for storing the string "PROGRAMMING" in C. (2)
- 4 What is the purpose of a switch statement? (2)
- 5 Give the differences between while and do-while statement (3)
- 6 What are formal arguments and actual arguments in a function? (2)
- 7 What are function prototypes? Why do we use function prototypes? (3)
- 8 How is an array name interpreted when it is passed to a function? (2)
- 9 How do you declare a pointer variable? What is the significance of the datatype included in the declaration? (3)
- 10 How do you interpret the following function declaration? int *p(char a[]) (2)
- 11 What is the purpose of typedef feature? (2)
- 12 What is union? How does it differ from a structure? (3)
- 13 What do you mean by opening of a file? How is this accomplished? (3)
- 14 What are enumeration constants? How ordinal values are assigned to them? (3)
- 15 Discuss the different parameters that are passed to main function as command line arguments. (3)
- 16 Explain register storage class with the help of an example. (2)

PART B*Answer any four questions. Each carries 8 marks.*

- 17 a) Write a C program to test whether a given number is palindrome or not. (5)
- b) Discuss the differences between break and continue statements in C. (3)
- 18 a) Write a C program to find the largest and smallest numbers and their locations in an array of n numbers. (5)
- b) Explain recursion with the help of an example. (3)
- 19 a) Write a C program to sort a set of numbers using bubble sort. (6)
- b) Discuss any four bit level operators with suitable examples. (2)
- 20 a) Write a C program to find the transpose of a matrix. (6)
- b) How can you access structure members using a pointer to structure variable? (2)
- 21 a) Write a C program to concatenate two strings without using any standard library function. (6)
- b) What is the use of an indirection operator? (2)

PART C

Answer any two questions. Each carries 14 marks.

- 22 a) Write a function to perform binary search on a set of sorted numbers. Write a complete C program which accepts a sorted array of N numbers and invokes the function to check for the presence of a particular key element in the array. (10)
- b) Compare formatted files and unformatted files (4)
- 23 a) Write a program to count the number of vowels, consonants, digits and special characters in a text file. (10)
- b) Discuss the different parameter passing techniques. (4)
- 24 a) A library database maintains following information about books:- book_id, name, author, no_of_copies. Write a program to sort the books based on the decreasing order of number of copies available. (10)
- b) What are array of pointers? How do you declare an array of pointers? (4)

Reg No.:		Name:	
APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY			
FOURTH SEMESTER B.TECH DEGREE EXAMINATION, JULY 2017			
Course Code: CS100			
Course Name: COMPUTER PROGRAMMING (CS, IT)			
Max. Marks: 100		Duration: 3 Hours	
PART A			
<i>Answer all Questions</i>			
1		Variable-1 mark , Declaration of variables -1	
2		x++ with example – 1.5 marks , ++x with example – 1.5 marks	
3		char str[] = "PROGRAMMING" or char str[12]= "PROGRAMMING" - 2 marks	
4		Purpose of switch statement – 2 marks	
5		Any three differences- 3 marks	
6		Formal arguments(1 mark) ,Actual argument (1 mark)	
7		function prototype(1.5 marks), use of function prototypes (1.5 marks)	
8		Array name is a pointer to the first element of the array, hence call by reference method of parameter passing happens (2 marks)	
9		Syntax of declaring a pointer variable (1.5 marks) Significance of the datatype(1.5 marks)	
10		P is a function which accepts a character array and returns a pointer to an integer variable(2 marks)	
11		typedef feature(1.5 marks) +example(0.5 marks)	
12		Union(1 mark) difference between structure and union(2 marks)	
13		opening of a file(1 mark) Explain fopen () and different modes - 2 marks	
14		enumeration constants(1 mark) example of ordinal value assignment(2 marks)	
15		Parameters – datatype and function of each parameter(3marks)	
16		Explain register storage class(1.5 marks),example(0.5 marks)	
PART B			
<i>Answer any 4 complete questions, each having 8 marks</i>			
17	a)	Complete program with no syntax errors(5 marks)	
	b)	differences between break and continue- any three (3 marks)	

18	a)	Finding largest number- 2 marks Finding smallest number- 2 marks Printing their positions-1 mark	
	b)	Explain Recursion(1.5 marks), example(1.5 marks)	
19	a)	Complete program with no syntax errors(6 marks)	
	b)	any four bit level operators with suitable examples(0.5 marks each)- 2 marks	
20	a)	Complete program with no syntax errors(6 marks)	
	b)	access structure members using a pointer to structure variable using -> operator + example(2 marks)	
21	a)	Complete program with no syntax errors(6 marks) (Concatenation by using library function (1.5 marks))	
	b)	use of indirection operator (*) with example (2 marks)	
PART C			
<i>Answer any two questions each having 14 marks</i>			
22	a)	Function for binary search (6 marks). Complete program with no syntax errors(10 marks)	
	b)	Compare formatted files and unformatted files(4 marks)	
23	a)	Complete program with no syntax errors(10 marks) (Opening and accessing text from file – 2 marks Each computation carries 2 marks each- (4*2)=8 marks)	
	b)	Call by value (2 marks), call by reference (2 marks)	
24	a)	Complete program with no syntax errors(10 marks) (Creating a library database by accepting information from user -4 marks Sorting based on number of copies and displaying result – 6 marks)	
	b)	Use of array of pointers(2 marks) declaration (2 marks)	

SCMS SCHOOL OF ENGINEERING AND TECHNOLOGY

VIDYA NAGAR, PALISSERY, KARUKUTTY.

Internal Test – I		Regulations - 2015	
Programme: B. Tech CSE	Semester: II	Max. Marks: 50	Duration: 2 Hrs
Course Code & Title:CS 100COMPUTER PROGRAMMING			
Batch: 2017	Class: 2CS1 & CS2	Date:9.02.2018	Time:2.00 PM to 4.00 PM
Knowledge Levels (KL)	K1 - Remembering	K3 - Applying	K5 – Evaluating
	K2 - Understanding	K4 – Analysing	K6 – Creating

Answer all questions

PART A

NO.	Question	Marks	CO	KL
1	Write a program to find the largest of three numbers using conditional operator	5	CO1	K3
2	What will be the output of the following code? Explain inti=-1,j=-1,k=0,l=2,m; m = i++ && j++&& k++ --l; printf("%d %d %d %d %d",i,j,k,l,m);	5	CO1	K5
3	Differentiate between entry controlled and exit controlled loop with example	5	CO1	K1
4	Explain various tokens in C	5	CO1	K1

PART B

Answer all questions

NO.	Question	Marks	CO	KL
1	Write a C program to display the count of prime numbers within a limit	10	CO1	K3
2	Write a menu driven C program to perform the following	10	CO1	K3

	(i) Factorial of a number. (ii) Generate Fibonacci series up to a limit			
3	a) Write a C program to check whether a given number is palindrome or not b) Discuss the differences between break and continue statements in C	10	CO1	K3

SCMS SCHOOL OF ENGINEERING & TECHNOLOGY Vidya Nagar,
Karukutty, Ernakulam-683582

Computer Science & Engineering
Semester II

COMPUTER PROGRAMMING

Internal Test – I

Feb 2018

Max Marks: 50

Time: 2Hr

Answer all questions

PART A

1. Write a program to find the largest of three numbers using conditional operator.

```
#include <stdio.h>

int main()
{
    int a, b, c, max;
    printf("Enter Three Integers\n");
    scanf("%d %d %d", &a, &b, &c);
    max = (a > b) ? ((a > c) ? a : c) : ((b > c) ? b : c);
    printf("Maximum Number is = %d\n", max);
    return 0;
}
```

2. What will be the output of the following code? Explain

```
int i=-1, j=-1, k=0, l=2, m;
m = i++ && j++ && k++ || --l;
printf("%d %d %d %d %d", i, j, k, l, m);
```

output: 0 0 1 1 1

3. Differentiate between entry controlled and exit controlled loop with example.

Entry Controlled Loop	Exit Controlled Loop
Test condition is checked first, and then loop body will be executed.	Loop body will be executed first, and then condition is checked.

If Test condition is false, loop body will not be executed.

If Test condition is false, loop body will be executed once.

for loop and while loop are the examples of Entry Controlled Loop.

do while loop is the example of Exit controlled loop.

Entry Controlled Loops are used when checking of test condition is mandatory before executing loop body.

Exit Controlled Loop is used when checking of test condition is mandatory after executing the loop body.

```
int count=100;
while(count<50)
printf("%d",count++);
```

```
int count=100;
do
{
printf("%d",count++);
}while(count<50);
```

4. Explain various tokens in C.

- C tokens are the basic buildings blocks in C language which are constructed together to write a C program.
- Each and every smallest individual units in a C program are known as C tokens. C tokens are of six types. They are,
 1. Keywords (eg: int, while),
 2. Identifiers (eg: main, total),
 3. Constants (eg: 10, 20),
 4. Strings (eg: "total", "hello"),
 5. Special symbols (eg: (), {}),
 6. Operators (eg: +, /, -, *)

[4*5=20Marks]

PART B

1. Write a C program to display the count of prime numbers within a limit.

```
#include <stdio.h>
void main()
{
inti, j, n, flag,p=0;
```



```

printf("Find prime numbers between 1 to : ");
scanf("%d", &n);
printf("All prime numbers between 1 to %d are:\n", n);
for(i=2; i<=n; i++)
{
    flag = 1;
    for(j=2; j<=i/2; j++)
    {
        if(i%j==0)
        {
            flag = 0;
            break;
        }
    }

    if(flag==1)
    {
        p=p+1;
        printf("%d, ", i);
    }
}printf("\n%d",p);
}

```

2. Write a menu driven C program to perform the following

(i) Factorial of a number.

}

(ii) Generate Fibonacci series up to a limit

```
#include <stdio.h>
```

```
intmain()
```

```
{
```

```
inti,f=1,num,ch,a, b, c, terms;
```

```
printf("1.factorial of a no 2.fibanocci series\n");
```

```
printf("Enter the choice");
```

```
scanf("%d",&ch);
```

```
switch(ch)
```

```
{
```

```
case 1:
```

```
printf("\nEnter a number: ");
```

```

scanf("%d",&num);
for(i=1;i<=num;i++)
{
    f=f*i;
}
printf("Factorial of %d is: %d",num,f);
break;
case 2:
printf("Enter number of terms: ");
scanf("%d", &terms);
a = 0;
b = 1;
c = 0;
printf("Fibonacci terms: \n");
for(i=1; i<=terms; i++)
{
printf("%d\t", c);
a = b;
b = c;
c = a + b;
}
break;
}
}

```

3. a) Write a C program to check whether a given number is palindrome or not

```

#include <stdio.h>
void main()
{
int n, rev = 0, r, num;
printf("Enter an integer: ");
scanf("%d", &n);
num= n;
while( n!=0 )
{
r = n%10;
rev = rev*10 + r;
n /= 10;
}
if (num == rev)

```

```

printf("%d is a palindrome.", num);
    else
printf("%d is not a palindrome.", num);
}

```

b) Discuss the differences between break and continue statements in C.

S.No.	break	continue
1.	break statement is used in switch and loops.	continue statement is used in loops only.
2.	When break is encountered the switch or loop execution is immediately stopped.	When continue is encountered, the statements after it are skipped and the loop control jump to next iteration.

```

#include<stdio.h>
voidmain()
{
    inti;
    for(i=0;i<5;++i)
    {
        if(i==3)
        break;
        printf("%d ",i);
    }
}

```

```

#include<stdio.h>
voidmain()
{
    inti;
    for(i=0;i<5;++i)
    {
        if(i==3)
        continue;
        printf("%d ",i);
    }
}

```

[3*10=30Marks]

SCMS SCHOOL OF ENGINEERING AND TECHNOLOGY
VIDYA NAGAR, PALISSERY, KARUKUTTY.

Internal Test – II		Regulations · 2015	
Programme: B. Tech CSE	Semester: II	Max. Marks: 50	Duration: 2 Hrs
Course Code & Title:CS 100COMPUTER PROGRAMMING			
Batch: 2017	Class: 2CS1 & CS2	Date:6 .03.2018	Time:2.00 PM to 4.00 PM
Knowledge Levels (KL)	K1 · Remembering	K3 · Applying	K5 – Evaluating
	K2 · Understanding	K4 – Analysing	K6 – Creating

Answer all questions

PART A

NO.	Question	Marks	CO	KL
1	Explain declaration and initialization of 1-d and 2-d arrays in C with example	4	CO2	K1
2	Explain function and its different categories	4	CO4	K1
3	Write a program to find largest and smallest element in an array	4	CO2	K3
4	Write a C program to sort a set of names	4	CO5	K3
5	Explain Different Parameter Passing Methods with example	4	CO4	K1

PART B

Answer all questions

NO.	Question	Mark s	CO	KL
1	Write a C program to find the transpose of a matrix and check whether the given matrix is symmetric or not	10	CO2	K3
2	Write a menu driven program to find the sum of	10	CO4	K3

	(i) Sine series (ii) (ii) Cosine Series (iii) (iii) Exponential series			
3	Write a menu driven program to display (i) Floyds Triangle (ii) Pascals Triangle	10	CO4	K3

**SCMS SCHOOL OF ENGINEERING & TECHNOLOGY Vidya Nagar,
Karukutty, Ernakulam-683582**

Computer Science & Engineering
Semester II
COMPUTER PROGRAMMING
Internal Test – II

March 2018

Max Marks: 50
Time: 2Hr

Answer all questions

PART A

1. Explain declaration and initialization of 1-d and 2-d arrays in C with example.

Declaration of an array:- We know that all the variables are declared before they are used in the program. Similarly, an array must be declared before it is used. During declaration, the size of the array has to be specified. The size used during declaration of the array informs the compiler to allocate and reserve the specified memory locations.

Syntax:- `data_type array_name[n];` where, n is the number of data items (or) index (or) dimension. 0 to (n-1) is the range of array.

Ex: `inta[5]; float x[10];`

Arrays can be initialized at the time of declaration when their initial values are known in advance. Array elements can be initialized with data items of type int, char etc.

Ex:- `int a[5]={10,15,1,3,20};`

An array consisting of two subscripts is known as two-dimensional array. These are often known as array of the array. In two dimensional arrays the array is divided into rows and columns. These are well suited to handle a table of data.

In 2-D array we can declare an array as :

Declaration:- Syntax: `data_type array_name[row_size][column_size];`

Ex:- `int arr[3][3];` where first index value shows the number of the rows and second index value shows the number of the columns in the array. Initializing two-dimensional arrays: Like the one-dimensional arrays, two-dimensional arrays may be initialized by following their declaration with a list of initial values enclosed in braces. Ex: `inta[2][3]={0,0,0,1,1,1};` initializes the elements of the first row to zero and the second row to one. The initialization is done row by row.

2. Explain function and its different categories.

function is a group of statements that together perform a task. Every C program has at least one function, which is **main()**, and all the most trivial programs can define additional functions.

A function **declaration** tells the compiler about a function's name, return type, and parameters. A function **definition** provides the actual body of the function.

A function depending on whether the arguments are present or not and whether a value is returned or not, may belong to one of following categories

1. Function with no return values, no arguments
2. Functions with arguments, no return values
3. Functions with arguments and return values
4. Functions with no arguments and return values.

3. Write a program to find largest and smallest element in an array.

```
#include <stdio.h>

int main()
{
    int arr[100], n, i, small, large;
    printf("Enter the number of elements you want to insert : ");
    scanf("%d", &n);
    for (i = 0; i < n; i++)
    {
        printf("Enter element %d : ", i + 1);
        scanf("%d", &arr[i]);
    }
    small = arr[0];
    large = arr[0];
    for (i = 1; i < n; i++)
    {
        if (arr[i] < small)
        {
            small = arr[i];
        }
        if (arr[i] > large)
        {
            large = arr[i];
        }
    }
}
```

```

    }
}
printf("\nLargest element is : %d", large);
printf("\nSmallest element is : %d", small);
return 0;
}

```

4. Write a C program to sort a set of names.

```

#include <stdio.h>
#include <conio.h>
#include <string.h>

void main()
{
    char name[10][8], Tname[10][8], temp[8];
    int i, j, N;

    clrscr();

    printf("Enter the value of Nn");
    scanf("%d", &N);

    printf("Enter %d namesn", N);
    for(i=0; i< N ; i++)
    {
        scanf("%s", name[i]);
        strcpy (Tname[i], name[i]);
    }

    for(i=0; i< N-1 ; i++)
    {
        for(j=i+1; j< N; j++)
        {
            if(strcmpi(name[i], name[j]) > 0)
            {
                strcpy(temp, name[i]);
                strcpy(name[i], name[j]);
                strcpy(name[j], temp);
            }
        }
    }

    printf("n-----n");
    printf("Input NamesSortednamesn");
    printf("-----n");
    for(i=0; i< N ; i++)
    {
        printf("%stt%sn", Tname[i], name[i]);
    }
    printf("-----n");
}

```

```
}
```

5. Explain Different Parameter Passing Methods with example.

Pass By Value: This method uses *in-mode* semantics. Changes made to formal parameter do not get transmitted back to the caller. Any modifications to the formal parameter variable inside the called function or method affect only the separate storage location and will not be reflected in the actual parameter in the calling environment. This method is also called as **call by value**.

```
// C program to illustrate
// call by value
#include <stdio.h>

void func(int a, int b)
{
    a += b;
    printf("In func, a = %d b = %d\n", a, b);
}

int main(void)
{
    int x = 5, y = 7;

    // Passing parameters
    func(x, y);
    printf("In main, x = %d y = %d\n", x, y);
    return 0;
}
```

Pass by reference(aliasing): This technique uses *in/out-mode* semantics. Changes made to formal parameter do get transmitted back to the caller through parameter passing. Any changes to the formal parameter are reflected in the actual parameter in the calling environment as formal parameter receives a reference (or pointer) to the actual data. This method is also called as **call by reference**. This method is efficient in both time and space.

```
// C program to illustrate
// call by reference
#include <stdio.h>
```



```

void swapnum(int* i, int* j)
{
int temp = *i;
    *i = *j;
    *j = temp;
}

int main(void)
{
int a = 10, b = 20;

    // passing parameters
swapnum(&a, &b);

printf("a is %d and b is %d\n", a, b);
    return 0;
}

```

[5*4=20Marks]

PART B

1. Write a C program to find the transpose of a matrix and check whether the given matrix is symmetric or not.

```

#include<stdio.h>

intmain()
{
intA[10][10];
intB[10][10];
inti,j,r, c,isSymmetric;
printf("Enter the limit");
scanf ("%d%d",&r,&c);
printf("Enter elements in matrix of size 3x3: \n");
for(i=0;i<r;i++)
{
for(j=0;j<c;j++)
{

```

```
scanf("%d",&A[i][j]);  
}  
}
```

```
for(i=0;i<r;i++)  
{  
for(j=0;j<c;j++)  
{  
B[i][j]= A[i][j];  
}  
}
```

```
isSymmetric=1;  
for(i=0;i<r;i++)  
{  
for(j=0;j<c;j++)  
{  
if(A[i][j]!= B[i][j])  
{  
isSymmetric=0;  
break;  
}  
}  
}
```

```
if(isSymmetric==1)  
{  
printf("\n\nThe given matrix is Symmetric matrix: \n\n");  
}  
else  
{  
printf("\n\nThe given matrix is not Symmetric matrix.");  
}  
}
```

2. Write a menu driven program to find the sum of (i) Sine series (ii) Cosine Series (iii)

Exponential series

```
#include<stdio.h>
```

```
#include<conio.h>
```

```
void main()
```

```

{
    inti,n,j,ch;
    float x,t,s,r;
    char c='y';
    clrscr();
    do
    {
        printf("\n1.SINE SERIES 2.COSINE SERIES 3.EXPONENTIAL SERIES");
printf("\n ENTER THE CHOICE");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1:
                printf("\nENTER THE LIMIT");
                scanf("%d",&n);
                printf("\nENTER THE VALUE OF x:");
                scanf("%f",&x);
                r=((x*3.1415)/180);
                t=r;
                s=r;
                i=2;
                for(j=2;j<=n;j++)
                {
                    t=(t*(-1)*r*r)/(i*(i+1));
                    s=s+t;
                    i=i+2;
                }

```



```
printf("\nSUM OF THE GIVEN SINE SERIES IS %4f",s);
```

```
break;
```

case 2:

```
printf("\nENTER THE LIMIT ");
```

```
scanf("%d",&n);
```

```
printf("\nENTER THE VALUE OF x:");
```

```
scanf("%f",&x);
```

```
r=((x*3.14)/180);
```

```
t=1;
```

```
s=1;
```

```
i=1;
```

```
for(j=2;j<=n;j++)
```

```
{
```

```
    t=(-1)*t*r*r/(i*(i+1));
```

```
    s=s+t;
```

```
    i=i+2;
```

```
}
```

```
printf("\n SUM OF THE COSINE SERIES IS %f",s);
```

```
break;
```

case 3:

```
printf("\nENTER THE LIMIT");
```

```
scanf("%d",&n);
```

```
printf("\nENTER THE VALUE OF x:");
```

```
scanf("%f",&x);
```

```
t=1;
```

```
s=1;
```

```
for(i=1;i<n;i++)
```

```

        {
            t=(t*x)/i;
            s=s+t;
        }
        printf("\nSUM OF EXPONENTIAL SERIES IS %f",s);
        break;
    default:
        printf("\n WRONG CHOICE");
    }
    printf("\n DO U WANT TO CONTINUE Y/N");
    scanf("%c",&c);
}
while(c=='y');
getch();
}

```

3. Write a menu driven program to display (i) Floyds Triangle (ii) Pascals Triangle

```

#include<stdio.h>
#include<conio.h>
void main()
{
    int rows, coef = 1, space, i, j;
    clrscr();
    printf("\n1. (1) Floyds Triangle (2i) Pascals Triangle ");
    printf("\n ENTER THE CHOICE");
    scanf("%d",&ch);

```

```

switch(ch)
{
    case 1:
        printf("Enter number of rows: ");
        scanf("%d",&rows);
        for(i=0; i<rows; i++)
        {
            for(space=1; space <= rows-i; space++)
                printf(" ");

            for(j=0; j <= i; j++)
            {
                if (j==0 || i==0)
                    coef = 1;
                else
                    coef = coef*(i-j+1)/j;

                printf("%4d", coef);
            }
            printf("\n");
        }
        break;
    case 2:

        printf("Enter number of rows: ");
        scanf("%d",&rows);

        for(i=1; i<= rows; i++)
        {
            for(j=1; j <= i; ++j)
            {
                printf("%d ", number);
                ++number;
            } printf("\n");
        }
}
Getch();
}

```


SCMS SCHOOL OF ENGINEERING & TECHNOLOGY, VIDYA NAGAR,
KARUKUTTY-683582

Department of Computer Science & Engineering
Second Semester

CS 100 COMPUTER PROGRAMMING

Assignment Question 1

C01

1. Which of the following expressions are true

a) $!(5+5 >= 10)$ b) $5+5 == 10 || 1+3 == 5$ c) $5 > 10 || 10 < 20 \&\& 3 < 5$ d) $10 != 15 \&\& !(10 < 20) || 15 > 30$

2. Write c assignment statements to evaluate the following equations

a) $area = \pi r^2 + 2\pi rh$ b) $torque = 2m_1m_2 g$ c) $side = \sqrt{(a^2 + b^2 - 2ab\cos x)}$

$m_1 + m_2$

d) $energy = mass[acceleration * height + velocity^2 / 2]$

3. What is the output of the following program?

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int x=5,y=10,z=10;
    clrscr();
    x=y=z;
    printf("%d",x);
    getch();
}
```

4. What is the output of the following program? Explain the output

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int x=10;;
    clrscr();
    if(x==10)
    {
        printf("true");
    }
    else
    {
        printf("false");
    }
    getch();
}
```

5. What is the output of the following printf statements?
- `printf("%d %c %f", 10, 'x', 1.23);`
 - `printf("%2d %c %4.2f", 1234, 'x', 1.23);`
 - `printf("%d \t %4.2f", 1234, 456);`
 - `printf("\t%08.2f\t", 1234);`
 - `printf("%d %d %d", 10, 20);`
6. In response to the input statement `scanf("%4d %*%d", &year, &code, &count);` the following data is keyed in 198837.45. what values does the computer assign to the variable year, code and count .
7. Find the errors if any in each of the following segments
- `if(x+y=z && y>0)`
`printf(" ")`
 - `if(code>1);`
`a=b+c;`
`else`
`a=0;`
 - `if((p<0)||(q<0))`
`printf(" sign is negative")`
8. Assume `x=10`, state whether the following logical expressions are true or false
- `x==10 && x>10 && !x`
 - `x==10 || x>10 && !x`
 - `x==10 && x>10 || !x`
 - `x==10 || x>10 || !x`
9. Assuming that `x=5, y=0` and `z=1`. What will be their values after executing the following code segments
- `if(x&& y)`
`x=10;`
`else`
`y=10;`
 - `if(x||y||z)`
`y=10;`
`else`
`z=0;`
 - `if(x)`
`if(y)`
`z=10;`
`else`
`z=0;`
 - `if(x==0||x&&y)`
`if(!y)`
`z=0;`

`else`
`y=1;`

10) What is the output of the following program?

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int m=1;
    clrscr();
    if(m==1)
    {
        printf("Delhi");
        if(m==2)
```

```

        printf("Chennai");
    else
        printf("Banglore");
    }
    else
    {
        printf("end");
    }
    getch();
}

```

11) What is the output of the following segment when executed.

```

int x=10,y=20;
if((x<y)||((x+5)>10))
    printf("%d",x);
else
    printf("%d",y);

```

12) Write a program that will read the value of x and evaluate the following function

```

{ 1 for x>0
Y= {0 for x=0
{ -1 for x<0

```

Using

a) nested if statements b) else if statements c) conditional operator

13) Write a program to display a) Floyd's triangle b) Pascal's triangle

Assignment



CP Assignments

[Red signature]

Submitted by

Rosemal Biju

CS II, 52

Roll NO - 22

Chapter-3 Review Questions

Q 3.5 which of the following expression are true?

(a) $!(5+5 > = 10)$

false

(b) $5+5 == 10 \parallel 1+3 == 5$

true

(c) $5 > 10 \parallel 10 < 20 \&\& 3 < 5$

true

(d) $10! = 15 \&\& !(10 < 20) \parallel 15 > 30$

false

Q 3.7 Write C assignment statements to evaluate the following equations.

(a) $\text{Area} = \pi r^2 + 2\pi r h$

#include <stdio.h>

#include <conio.h>

void main()

{

int r, h;

float = area;



```

clearscreen();
printf("Enter the number:");
scanf("%d %d", &r, &h);
area = (3.14 * r * r * r + 2 * 3.14 * r * h);
printf("%f", area);
getch();
}

```

(b)
$$G_{\text{torque}} = \frac{2m_1 m_2}{m_1 + m_2} \cdot g$$

```

#include <stdio.h>
#include <conio.h>
void main ()
{
    int m1, m2, x, y;
    float t;
    clearscreen();
    printf("Enter the value of m1, m2 |n");
    scanf("%d %d", &m1, &m2);
    x = (2 * m1 * m2 * 9.8);
    y = (m1 + m2)

```



```
t = x/y;  
printf("%f", t);  
getch();  
}
```

$$(c) \text{ Side} = \sqrt{a^2 + b^2 - 2ab \cos \alpha}$$

```
#include <stdio.h>
```

```
#include <conio.h>
```

```
#include <math.h>
```

```
#define cos x
```

```
void main()
```

```
{
```

```
int a, b, p, n;
```

```
float side;
```

```
clrscr();
```

```
printf("Enter the value of a, b, x\n");
```

```
scanf("%d %d %d", &a, &b, &x);
```

```
p = (a*a + b*b) - (2*a*b);
```

```
n = cos(x);
```

```
side = sqrt(p*n);
```

```
printf("%f", side);
```

```
getch();
```

```
}
```

(d) Energy = mass $\left[\text{acceleration} \times \text{height} + \frac{(\text{velocity})^2}{2} \right]$

```
#include <stdio.h>
```

```
#include <conio.h>
```

```
#include <math.h>
```

```
void main()
```

```
{
```

```
int m, h, v;
```

```
float energy;
```

```
clrscr();
```

```
acceleration = 9.8;
```

```
printf("Enter the values:");
```

```
scanf("%d %d %d", &m, &h, &v);
```

```
x = (9.8 * h);
```

```
y = (v * v) / 2;
```

```
energy = m * (x + y);
```

```
printf("%f", energy);
```

```
getch();
```

```
}
```

Q 3.13 What is printed by following program?

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int x = 5, y = 10, z = 10;
    clrscr();
    x = y == z;
    printf("%d", x);
    getch();
}
```

Output : 1

Q 3.16 What is the output of the following program? Explain the output

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int x = 10; y = 10, z = 10;
    if (x == 20)
    {
```



```
printf("true");  
}  
else  
{  
printf("false");  
}  
getch();  
}
```

Output : true

Q 3.20 What do the output of the following segment when executed

```
int m = -14, n = 3;
```

```
printf("%d\n", m/n * 10);
```

```
n = -n;
```

```
printf("%d\n", m/n * 10);
```

Output : -40

40

Chapter - 4

Q 4.5 State the output produced by the following printf statements

(a) `printf ("%d %c %f", 10, 'x', 1.23);`

output 10 x 1.230000

(b) `printf ("%d %c %4.2f", 1234, 'x', 123);`

output : error (2 commas) otherwise 1234 x 1.23

(c) `printf ("%d\t %4.2f", 1234, 456);`

output

(d) `printf ("%08.2f \\", 1234);`

output : " 00123.40"

(e) `printf ("%d %d %d", 10, 20);`

output : 10 20

Q 4.10 In response to the input statement

`scanf ("%4d %*s", &year, &code, &count);`

the following data is keyed in :

19883745

What values does the computer assign to the variable year, code and count?

Output - 1988

Chapter - 5

Q 5.3 Find the errors, if any in each of the following statements

(a) `if (x+y = z && y > 0)`

`printf (" ");`

Ans: Error

Soln: `if (x+y == z) && (y > 0)`

`printf (" ");`

(b) `if (code > 1)`

`a = b + c`

`else`

`a = 0`

Output: error

Sol: `if (code > 1)`

`a = b + c;`

`else`

`a = 0;`

(c) `if (p < 0) || (q < 0)`

`printf ("sign is negative");`

Ans: error

Sol: `if ((p > 0) || (q < 0))`

`printf ("sign is negative");`

5.6 Assuming $x = 10$, state whether the following logical expressions are true or false

(a) $x == 10 \ \&\& \ x > 10 \ \&\& \ !x$

output: False

(b) $x == 10 \ \|\| \ x > 10 \ \&\& \ !x$

output: true

(c) $x == 10 \ \&\& \ x > 10 \ \|\| \ !x$

output: False

(d) $x == 10 \ \|\| \ x > 10 \ \|\| \ !x$

output: true

5.9 Assuming that $x = 5$, $y = 0$, and $z = 1$ initially, what will be their values after executing the following code statements.

(a) $if \ (x \ \&\& \ y)$

$x = 10;$

else

$y = 10;$

Output

10

10

(b) if (x || y || z)

y = 10;

else

x = 0;

output

1
0

(c) if (x)

if (y)

z = 10;

else

z = 0;

output

10
0

(d) if (x == 0 || x << y)

if (!y)

z = 0;

else

y = 1;

output

0
1

Q 5.13 What is the output of the following program?

main()

```
{
```

```
int m = 1;
```

```
if (m == 1)
```

```
{
```

```
printf("Delhi");
```

```
if (m == 2)
```

```
printf("Chennai");
```

```
else
```

```
printf("Bangalore");
```

```
}
```

```
else:
```

```
printf("END");
```

```
}
```

output

1

Delhi

2

Chennai

3

Bangalore.

Q 5.19 what is the output of the following segment when executed?

```
int x = 10, y = 20;
```

```
if ((x < y) || (x + 5) > 10)
```

```
printf("%d", x);
```


else

```
printf("%d", y);
```

output

10

Programming Exercise

Q 5.9 Write a program that will read the value of x and evaluate the following function.

$$y = \begin{cases} 1 & \text{for } x > 0 \\ 0 & \text{for } x = 0 \\ -1 & \text{for } x < 0 \end{cases}$$

using

(a) nested if statements

Sol:

```
#include <stdio.h>
```

```
#include <conio.h>
```

```
void main()
```

```
{
```

```
float x, y;
```

```
clrscr();
```

```
printf("Input x\n");
```

```
scanf("%f", &x);
```

```
if (x != 0)
```

```
{
```

```
if (x > 0)
```

```
printf("y = 1");
```

```
if (x < 0)
```

```
printf("y = -1");
```

```
}
```

```
if (x == 0)
```

```
printf("y = 0");
```

```
getch();
```

```
}
```

(b) else if statements

```
#include <stdio.h>
```

```
#include <conio.h>
```

```
void main()
```

```
{
```

```
float x, y;
```

```
clrscr();
```

```
printf("input x\n");
```

```
scanf ("%f", &x);
```

```
if (x == 0)
```

```
{
```

```
if (x > 0)
```

```
{
```

```
printf ("1");
```

```
}
```

```
else
```

```
printf ("-1");
```

```
}
```

```
else
```

```
printf ("0");
```

```
getch();
```

```
}
```

(c) Conditional Operator

```
#include <stdio.h>
```

```
#include <conio.h>
```

```
void main()
```

```
{
```

```
float y, x;
```

```
clrscr();
```



```
printf ("Input x\n");
```

```
scanf ("%f", &x);
```

```
y = (x != 0) ? ((x > 0) ? 1 : -1) : 0;
```

```
printf ("%d", y);
```

```
getch();
```

```
}
```

Q 5.12 An electricity board charges the following the rates for the use of electricity:

For the first 200 units : 80p per unit

For the next 100 units : 90p per unit

Beyond 300 units : Rs 1.00 per unit

All users are charged a minimum of Rs. 100 as meter charge. If the total amount is more than Rs. 400,

then an additional subcharge of 15% of total

amount is charged write a program to record

the names of users and number of units consumed

and print out the charges with names.

```
# include <stdio.h>
```

```
# include <conio.h>
```

```
void main()
```

```
{
```

```
float units, total, net;
```

```
char name;
```

```
clrscr();
```

```
printf ("Input your name and units\n");
```

```
scanf ("%s %f", &name, &units);
```

```
{
```

```
if (units <= 200)
```

```
total = 100 + 0.80 * units;
```

```
else if (units <= 300)
```

```
total = 100 + 0.90 * units;
```

```
else if (units > 300)
```

```
total = 100 + 1.00 * units;
```

```
}
```

```
if (total > 400)
```

```
{
```

```
net = total + total * 0.15;
```

```

printf ("total = %f", net);
}
else
printf ("total = %f", total);
getch();
}

```

Q Write a C program to display pascal's triangle

```

#include <stdio.h>
#include <conio.h>
void main()
{
int rows, coefficient = 1, space, i, j;
clrscr();
printf ("Enter number of rows:");
scanf ("%d", &rows);
for (i = 0; i < rows; i++)
{
for (space = 1; space <= rows - i; space++)
printf (" ");
for (j = 0; j <= i; j++)

```



```
if (j == 0 || i == 0)
```

```
    coefficient = 1;
```

```
else
```

```
    coefficient = coefficient (i - j + 1) / j;
```

```
    printf ("%d", coefficient);
```

```
}
```

```
printf ("\n");
```

```
}
```

```
return 0;
```

```
}
```

Q Write a C program to display Floyd's triangle

```
#include <stdio.h>
```

```
#include <conio.h>
```

```
void main()
```

```
{
```

```
    int row, i, j, number = 1;
```

```
    clrscr();
```

```
    printf ("enter number of rows:");
```

```
scanf ("%d", &rows);
```

```
for (i = 1 ; i <= rows; i++)
```

```
{
```

```
for (j = 1 ; j <= i ; ++j)
```

```
{
```

```
printf ("%d", numbers);
```

```
++number;
```

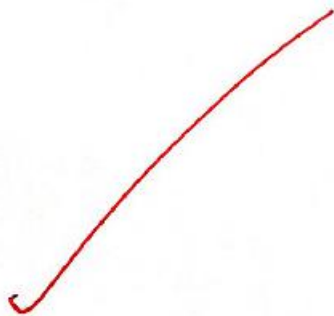
```
}
```

```
printf ("\n");
```

```
}
```

```
return 0;
```

```
}
```



CS

ASSIGNMENT

Handwritten signature in red ink.

Submitted by ,

Mariya Raphael
CS-II
Roll no. 3

1) Which of the following expressions are true?

a) $!(5+5 > = 10)$

b) $5+5 == 10 \parallel 1+3 == 5$

c) $5 > 10 \parallel 10 < 20 \&\& 3 < 5$

d) $10! = 15 \&\& !(10 < 20) \parallel 15 > 30$

Ans: - • $5+5 == 10 \parallel 1+3 == 5$ and

• $5 > 10 \parallel 10 < 20 \&\& 3 < 5$

are true expressions. They are always true.

2) Write C assignment statements to evaluate the following equations:

a) $\text{Area} = \pi r^2 + 2\pi r h$

b) $\text{Torque} = \frac{2m_1 m_2}{m_1 + m_2} \cdot g$

c) $\text{Side} = \sqrt{a^2 + b^2 - 2ab \cos(\alpha)}$

d) $\text{Energy} = \text{mass} \left[\text{acceleration} \times \text{height} + \frac{(\text{velocity})^2}{2} \right]$

Ans: ~~(a) Area =~~

a) Area

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
#define PI 3.141592
void main()
{
    int area r, h;
    float area;
    clrscr();
    printf("Enter the value for r and h");
    scanf("%d %d", &r, &h);
    area = PI * pow(r, 2) + 2 * PI * r * h;
    printf("The area is %f", area);
    getch();
}
```

b) Torque

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int m1 m1, m2;
    float torque;
```



```

clear();
printf("Enter the values of mass1 and mass2");
scanf("%d %d", &m1, &m2);
torque = (2 * m1 * m2) / (m1 + m2) * 9.8;
printf("%f", torque);
getch();
}

```

9) Side

```

#include <stdio.h>
#include <conio.h>
#include <math.h>
void main()
{
int a, b, x;
float side;
clear();
printf("Enter the values of a and b");
scanf("%d %d", &a, &b);
printf("Enter the value for angle x");
scanf("%d", &x);
m = 2 * a * b * cos(x);
n = pow(a, 2) + pow(b, 2) - m

```



```
side = sqrt(n);  
printf("%f", side);  
getch();  
}
```

d) Energy

```
#include <stdio.h>  
#include <conio.h>  
#include <math.h>  
void main()  
{  
int mass, acceleration, height, velocity;  
float energy, v;  
clrscr();  
printf("Enter mass, acceleration, height and  
velocity");  
scanf("%d %d %d %d", &mass, &acceleration,  
&height, &velocity);  
v = pow(velocity, 2) / 2;  
energy = (acceleration * height + v) * mass;  
printf("%f", energy);  
getch();  
}
```

to 20 and seems to be true in condition checking and execute the corresponding block - TRUE.

5) What is the output ?

```
int m = -14, n = 3;  
printf ("%d \n", m/n * 10);  
n = -n;  
printf ("%d \n", m/n * 10);
```

Ans:-

Output :-
-40
40

Because m and n are integer type. So it neglects the fractional part and do the calculation only with the integer part.

6) State the outputs produced by the following printf statements.

a) `printf ("%d %c %f", 10, 'x', 1.23)`

Ans: 10 X 1.230000

b) `printf ("%2d %c %4.2f", 1234, 'x', 1.23)`

Ans:

• 12 X 1.23

one space

c) `printf("%d\t%4.2f", 1234, 456);`

Ans: 1234

4

d) `printf("\%08.2f\%", 123.4);`

e) `printf("%d %d %d", 10, 20);`

7) In response to the input statement,
`scanf("%4d %* %d", &year, &code, &count);`
the following data is keyed in:

19883745

8) Find errors in following segments.

a) `if (x+y = z && y > 0)`
`printf(" ");`

Ans: `if ((x+y == z) && (y > 0))`
~~`printf(" ");`~~

b) ~~`if (p < 0) || (q < 0)`~~

`printf(" sign is negative");`

Ans: if (p < 0) || (q < 0)

printf("sign is negative");

c) if (code > 1);

a = b + c

else

a = 0

Ans: if (code > 1) · if (code > 1)

a = b + c;

a = b + c;

else

else

a = 0;

a = 0;

d
9) Assume that x = 5, y = 0 and z = 1 initially.
If so, what will be the outputs?

a) if (x < y)

x = 10;

else

y = 10;

b) if (x || y || z)

y = 10;

else

z = 0;

```
c) if (x)
    if (y)
        z = 10;
    else
        z = 0;
```

```
d) if (x == 0 || x << y)
    if (!y)
        z = 0;
    else
        y = 1;
```

10) what is the output of following code ?

```
main ()
{
    int m = 1
    if (m == 1)
    {
        printf ("Delhi");
        if (m == 2)
            printf ("chennai");
        else
```

```

else pr
printf (" Bangalore");
}
else;
Printf (" END");
}

```

Output :- Delhi

1) what will be the output of following code ?

```

int x = 10, y = 20 ;
if ((x < y) || (x + 5) > 10)
printf ("%d ", x);
else
printf ("%d ", y);

```

Ans: Output :- 10

2) Write a program that will read value of x and evaluate following function.

$$y = \begin{cases} 1 & \text{for } x < 0 \\ 0 & \text{for } x = 0 \\ -1 & \text{for } x > 0 \end{cases}$$

using

- a) nested if statements
- b) else if statements
- c) conditional operator ?!

Ans: Else-if statements

```
# include <stdio.h>
# include <conio.h>

void main ()
{
    int y, x;
    clrscr();
    printf("Enter the value of x");
    scanf("%d", &x);
    if (x > 0)
    {
        y = 1;
        printf("The value of y for given value %d is %d",
               x, y);
    }
    elseif (x == 0)
    {
        y = 0;
        printf("value of y for given value %d is %d", x, y);
    }
}
```

```
else  
{ y = -1
```

```
printf("value of y for given value %d is %c  
x, y");  
}
```

```
getch();
```

```
}
```

Output :-

Enter the value of x 29

The value of y for given value 29 is 1

Nested if Statements

```
#include <stdio.h>
```

```
#include <conio.h>
```

```
void main ()
```

```
{
```

```
int x, y;
```

```
clrscr();
```

```
printf("Enter the value of x");
```

```
scanf("%d", &x);
```

```
if
```



if (x > 0)

```
{  
    y = 1 ;  
    printf("value of y is %d", y);  
}
```

else

```
{  
    if (x == 0)  
    {  
        y = 0 ;  
        printf("value of y is %d", y);  
    }  
}
```

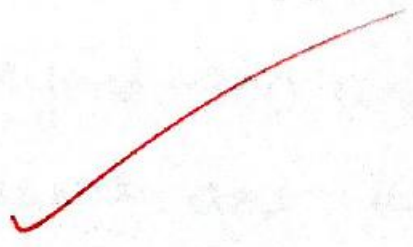
else

```
{  
    y = -1 ;  
    printf("value of y is %d", y);  
}
```

}

getch();

}



Conditional Operator

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int x, y;
    clrscr();
    printf("Enter the value of x");
    scanf("%d", &x);
    (x > 0) ? (y = 1) : (x == 0) ? x (y = 0) : (y = -1);
    printf("The value of y for given value %d is %d", x, y);
    getch();
}
```

13) An electricity board charges the following rates.

- For first 200 units : 80 P per unit
- For next 100 units : 90 P per unit
- Beyond 300 units : Rs 1.00 per unit

All users are charged a minimum of Rs 100 as meter charge. If the total amount is more than Rs 400, then an additional surcharge of 15% of total amount is charged. Write a program to

Read the names of users and number of units consumed and print out the charges with names.

Ans: -

```
#include <stdio.h>
#include <conio.h>
void main ()
{
    int units ;
    float charge ;
    char name [20] ;
    clrscr() ;
    printf ("Enter the name" );
    scanf ("%s", &name);
    printf ("Enter the total units of electricity used");
    scanf ("%d", &unit);
    if (units > 0 && unit <= 200)
    {
        charge = 100 + (units * 0.80);
    }
    elseif
    else if (units <= 300)
    {
        charge = 100 + (unit * 0.90);
    }
}
```



```
else  
{  
    charge = 100 + unit ;  
}
```

```
if (charge > 400)
```

```
{  
    charge = charge + (charge * 15) ;
```

```
    printf ("The the electricity charge, including surcharge for  
            %s is %f Rs" , name, charge);
```

```
}
```

```
else
```

```
{
```

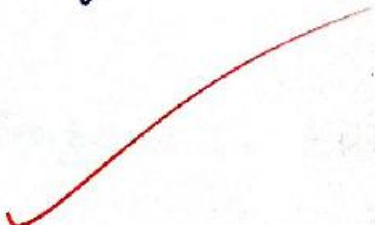
```
    printf ("The electricity charge for %s  
            is %f Rs" . name, charge);
```

```
}
```

```
getch();
```

```
}
```

14) Write a C program to implement Floyd's triangle.



FLOYD'S TRIANGLE

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int n, i, j, k = 1;
    clrscr();
    printf("Enter the number of rows");
    scanf("%d", &n);
    k = 1;
    for(i = 1; i <= n; i++)
    {
        printf("\n");
        for(j = 1; j <= i; j++)
        {
            printf("%d\t", k);
            k = k + 1;
        }
    }
    getch();
}
```

Output :- Enter number of rows 3

```
1
2 3
4 5 6
```

write program to implement pascal's triangle

```
Ans. #include <stdio.h>
#include <conio.h>
void main ()
{
    int x, y, n, a, z, s;
    printf ("Enter the no. rows ");
    scanf ("%d", &n);
    printf ("\n\n");
    s = n;
    for (x = 0; x <= n; x++)
    {
        a = 1;
        for (z = 0; z <= x; z++)
        {
            printf (" ");
            s--;
            for (y = 0; y <= z; y++)
            {
                printf ("%d", a);
                a = (a * (x - y) / (y + 1));
            }
            printf ("\n");
        }
        getch();
    }
}
```


Write a program to display cosine and sine series

Ans: #include <stdio.h>

#include <conio.h>

void main()

{

int i, n, j, ch;

float x, t, s, r;

choice

printf("1. Sine series \n 2. Cosine series");

printf("Enter your choice");

scanf("%d", &ch);

switch(ch)

{

case 1:

printf("Enter the limit");

scanf("%d", &n);

printf("\n Enter the value of x");

scanf("%f", &x);

r = ((x * 3.1415) / 180);

t = r;

s = r;

i = 2;

for(j=2; j<=n; j++)

{

t = (t * (-1) * r * r) / (i * (i+1));

s = s + t;

i = i + 2;

}


```
printf("\n sum of sine series is %f", s);  
break;
```

Case 2:

```
printf("\n Enter the limit");
```

```
scanf("%d", &n);
```

```
printf("\n Enter the value of x");
```

```
scanf("%f", &x);
```

```
t = 1;
```

```
s = 1;
```

```
i = 1;
```

```
for (j = 2; j <= n; j++)
```

```
{
```

```
    t = ((-1) * t * x * x) / (i * (i + 1));
```

```
    s = s + t;
```

```
    i = i + 2;
```

```
}
```

```
printf("\n sum of cosine series is %f", s);
```

```
break;
```

default:

```
printf("Invalid choice);
```

```
getch();
```

```
}
```



SCMS SCHOOL OF ENGINEERING & TECHNOLOGY, VIDYA NAGAR,
KARUKUTTY - 683582

Department of Computer Science & Engineering
Semester 2

CS100 COMPUTER PROGRAMMING

Assignment Question 2

1. write a program that uses a function to sort an array of integers CO4
2. write a program for matrix addition using pointers CO3

ASSIGNMENT



Submitted by,

Mariya Raphael

S2, CS2

Roll no: 3

Write a program that uses a function to sort an array of integers.

Ans: Function - Sort - Array

```
#include <stdio.h>
#include <conio.h>
void sort (int m, int x[]);
void main ()
{
    int i;
    int marks[5] = {40, 90, 73, 81, 35};
    printf("Marks before sorting\n");
    for(i=0; i<5; i++)
    {
        printf("%d\t", marks[i]);
    }
    printf("\n\n");
    sort(5, marks);
    printf("Marks after sorting\n");
    for(i=0; i<5; i++)
    {
        printf("%d\t", marks[i]);
    }
    printf("\n\n");
    getch();
}
```

```
void sort (int m, int x[])
{
    int i, j, t;
    for(i=1; i<=m-1; i++)
    {
        for(j=1; j<=m-i; j++)
        {
```

```

if (x[j-1] >= x[j])
{
    t = x[j-1];
    x[j-1] = x[j];
    x[j] = t;
}
}
}

```

Output

Marks before sorting

40 90 73 81 35

Marks after sorting

35 40 73 81 90

2) Write a C program for matrix addition using pointer.

Ans: #include <stdio.h>

#include <conio.h>

void main ()

{

int a[5][5], b[5][5], c[5][5], i, j, m, n;

printf("Enter row size and column size");

scanf("%d %d", &m, &n);

printf("Enter first matrix");

```

for(i=0; i<m; i++)
{
for(j=0; j<n; j++)
{
scanf("%d", (*(a+i)+j));
}
}

```

```

printf("Enter the second matrix");

```

```

for(i=0; i<m; i++)
{
for(j=0; j<n; j++)
{
scanf("%d", (*(b+i)+j));
}
}

```

```

printf("Addition");

```

```

for(i=0; i<m; i++)
{
for(j=0; j<n; j++)
{
*(*(c+i)+j) = (*(a+i)+j) + (*(b+i)+j);
printf("%d", (*(c+i)+j));
}
printf("\n");
}
}

```

Output

Enter row size and column size

3 3

Enter first Matrix

1	2	3
4	5	6
7	8	9

Enter the second matrix

9	8	7
6	5	4
3	2	1

Addition

10	10	10
10	10	10
10	10	10

$$((i+(i+d)*c) * a) + ((i+(i+o)*c) * b) = ((i+(i+c)*c) * x)$$

$$((i+(i+c)*c) * x) + ((i+(i+o)*c) * b) = ((i+(i+c)*c) * x)$$

$$((i+(i+c)*c) * x)$$

E. P. ...

ASSIGNMENT

Submitted By,
Shallet Mary T Eldho

SE, CS2

Roll no: 30



1. Write a program that uses a function to sort an array of integers

Function - sort - Array

```
* Include <stdio.h>
```

```
* Include <conio.h>
```

```
void sort (int m, int x[])
```

```
void main()
```

```
{  
  int i;
```

```
  int marks[5] = {40, 90, 73, 81, 35};
```

```
  printf("Marks before sorting");
```

```
  for (i=0; i<5; i++)
```

```
  {  
    printf("%d", marks[i]);
```

```
  }
```

```
  printf("\n\n");
```

```
  sort(5, marks);
```

```
  printf("Marks after sorting\n");
```

```
  for (i=0; i<5; i++)
```

```
  {  
    printf("%d\t", marks[i]);
```

```
  }
```

```
  printf("\n\n");
```

```
  getch();
```

```
}
```

```
void sort (int m, int x[])
```

```
{  
  int i, j, t;
```

```
  for (i=1; i<=m-1; i++)
```

```
  {  
    for (j=1; j<=m-i; j++)
```

```
    {  
      if (x[j-1] > x[j])
```

```
      {
```



```

    t = x[j-1];
    x[j-1] = x[j];
    x[j] = t;
}
}
}

```

Output

Marks before sorting

40 90 73 81 35

Marks after sorting

35 40 73 81 90

2. write a C program for matrix addition using pointers

```

#include <stdio.h>

```

```

#include <conio.h>

```

```

void main()

```

```

{

```

```

    int a[5][5], b[5][5], c[5][5], e, j, m, n;

```

```

    printf("Enter row size and column size");

```

```

    scanf("%d %d", &m, &n);

```

```

    printf("Enter first matrix");

```

```

    for (i=0; i<m; i++)

```

```

    {
        for (j=0; j<n; j++)

```

```

        {

```

```

            scanf("%d", (*(a+i)+j));

```

```

        }

```

```

    }

```

```

    printf("Enter the second matrix");

```

```

for (i=0; i<m; i++)
{
    for (j=0; j<n; j++)
    {
        scanf ("%d", (*(b+i)+j));
    }
}

```

```

printf ("Addition");

```

```

for (i=0; i<m; i++)
{
    for (j=0; j<n; j++)
    {
        *(*(c+i)+j) = *(*(a+i)+j) + *(*(b+i)+j);
        printf ("%d", *(*(c+i)+j));
    }
    printf ("\n");
}
}

```

Output

Enter row size and column size

3 3

Enter first matrix

1 2 3

4 5 6

7 8 9

Enter second matrix

9 8 7

6 5 4

3 2 1

Addition

10 10 10

10 10 10

10 10 10

1) Write a program using pointers to compute the sum of all elements stored in an array.

Ans:

```
#include <stdio.h>
#include <conio.h>
void main ()
{
    int *p, sum=0;
    int x[5] = {5, 9, 6, 3, 7};
    i=0;
    p=x;
    while (i<5)
    {
        sum = sum + *p;
        i++;
        p++;
    }
    printf ("In Sum = %d", sum);
    getch();
}
```



Output

Sum = 55

2) Write a function using pointers to exchange the values stored in two locations in the memory.

Ans:

```
#include <stdio.h>
#include <conio.h>
void change (int *, int *);
```



```
void main ()
```

```
{
```

```
int x, y;  
x = 100;  
y = 20;  
printf("Before exchange :");  
printf("x = %d y = %d \n", x, y);  
change (&x, &y);  
printf("After exchange :");  
printf("x = %d y = %d \n", x, y);  
getch();
```

```
}
```

```
void exchange (int *a, int *b)
```

```
{
```

```
int t;  
t = *a;  
*a = *b;  
*b = t;
```

```
}
```

Output

Before exchange : x = 100 y = 20
After exchange : x = 20 y = 100

1. Write a program using pointers to compute the sum of all elements stored on an array.

```
* Include <stdio.h>
```

```
* Include <conio.h>
```

```
void main()
```

```
{
```

```
int *p, sum = 0, i;
```

```
int x[5] = {5, 9, 6, 3, 7}
```

```
p = 0;
```

```
p = x;
```

```
while (p < 5)
```

```
{
```

```
sum = sum + *p;
```

```
p++;
```

```
p++;
```

```
}
```

```
printf ("\n sum = %d", sum);
```

```
getch();
```

```
}
```



Output

sum = 55

2. Write a function using pointers to exchange the values stored on two locations on the memory.

```
* Include <stdio.h>
```

```
* Include <conio.h>
```

```
void change (int *, int *);
```

```
void main()
```

```
{
```

```
int x, y;
```

```
x = 10;
```

```
y = 20;
```

```
printf("Before exchange");
printf("x= %d y= %d \n", x, y);
change(&x, &y);
printf("After exchange");
printf("x= %d y= %d \n", x, y);
getch();
}
void change(int *a, int *b)
{
    int t;
    t = *a;
    *a = *b;
    *b = t;
}
```

Output

Before exchange : x=10 y=20

After exchange : x=20 y=10

Plan text :

Write a program using pointer to compute the sum of all elements stored in an array.

* include <stdio.h>

* include <conio.h>

void main()

```
{
  int *p, sum = 0, i;
```

```
  int a[5] = {5, 9, 6, 3, 7};
```

```
  while (i < 5)
```

```
  {
    sum = sum + *p;
```

```
    p++;
```

```
  }
  printf("The sum = %d", sum);
```

```
  getch();
```

Output

sum = 45



CLASS TEST - 1

1. Write a program using pointers to compute the sum of all elements stored in an array.

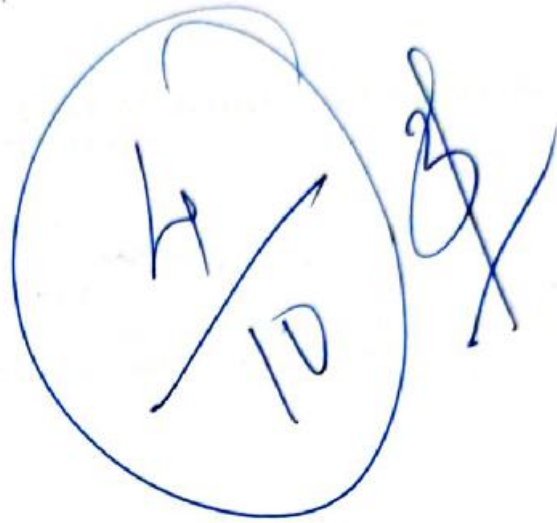
```

#include <stdio.h>
#include <conio.h>
void main()
{
    int *p, sum=0;
    int x[] = {5, 8, 7, 6, 4};
    i=0;
    p=x;
    while (i<5)
    {
        sum = sum + *p;
        i++;
        p++;
    }
    printf ("Sum = %d", sum);
    getch();
}

```

Output

Sum = 55



series
id

**SCMS SCHOOL OF ENGINEERING & TECHNOLOGY, VIDYA NAGAR,
KARUKUTTY-683582**

Department of Computer Science & Engineering

Semester 2

CS100 COMPUTER PROGRAMMING

ClassTest2

1. A file named "data" contains a series of integer numbers. Code a program to read these numbers and write all even numbers to a file named "even" and all odd numbers to a file named "odd".

C06

b

CLASS TEST-2

Mariya Raphael

S2, CS2

1) A file named DATA contains a series of integer numbers. Code a program to read these numbers and then write all 'odd' numbers to a file to be called ODD and all even numbers to a file EVEN.

```
Ans: #include <stdio.h>
void main ()
{
    FILE *f1, *f2, *f3;
    int number, i;
    f1 = fopen("DATA", "w");
    for(i=1; i<=30; i++)
    {
        scanf("%d", &number);
        if (number == -1)
            break;
        putw(number, f1);
    }
    fclose(f1);
    f1 = fopen("DATA", "r");
    f2 = fopen("ODD", "w");
    f3 = fopen("EVEN", "w");
    while ((number = getw(f1)) != EOF)
    {
        if (number % 2 == 0)
            putw(number, f3);
        else
            putw(number, f2);
    }
}
```



```
fclose (f1);
fclose (f2);
fclose (f3);
f2 = fopen ("ODD", "r");
f3 = fopen ("EVEN", "r");
printf ("ODD file \n");
while ((number = getw (f2)) != EOF)
{
    printf ("%d\t", number);
}
printf ("EVEN file \n");
while ((number = getw (f3)) != EOF)
{
    printf ("%d\t", number);
}
fclose (f2);
fclose (f3);
}
```

Output

1 2 3 4 5 6 7 8 9 0

ODD file

1 3 5 7 9

EVEN file

2 4 6 8 0

1. A file named DATA contains a series of Integer numbers. code a program to read these numbers and then write all odd numbers to a file to be called ODD and all even numbers to a file EVEN.

```
*include <stdio.h>
```

```
void main()
```

```
{
```

```
FILE *f1, *f2, *f3;
```

```
int number, e;
```

```
f1 = fopen("DATA", "w");
```

```
for (e=1; e<=30; e++)
```

```
{ scanf("%d", &number);
```

```
if (number == -1)
```

```
break;
```

```
putw(number, f1);
```

```
}
```

```
fclose(f1);
```

```
f1 = fopen("DATA", "r");
```

```
f2 = fopen("ODD", "w");
```

```
f3 = fopen("EVEN", "w");
```

```
while ((number = getw(f1)) != EOF)
```

```
{ if (number % 2 == 0)
```

```
putw(number, f3);
```

```
else
```

```
putw(number, f2);
```

```
}
```




```

fclose (f1);
fclose (f2);
fclose (f3);
f2 = fopen ("ODD", "w");
f3 = fopen ("EVEN", "w");
printf ("ODD file\n");
while ((number = getw (f2)) != EOF)
{
    printf ("%d\t", number);
}
printf ("Even file\n");
while ((number = getw (f3)) != EOF)
{
    printf ("%d\t", number);
}
fclose (f2);
fclose (f3);
}

```

Output

1 2 3 4 5 6 7 8 9 0

ODD file

1 3 5 7 9

EVEN file

2 4 6 8 0

1. A file named data contains a series of integer numbers. Code a program to read these numbers and then write all 'odd' numbers to a file to be called odd and all even numbers to a file even.

```

1: #include <stdio.h>
   #include <conio.h>
   ⚡
   file *f1, *f2, *f3;
   int num;
   f1 = fopen("data", "w");
   for(i=0; i<30; i++)
   ⚡
     scanf("%d", &num);
     if (num == -1)
       break;
     putw(num, f1);
   ⚡
   fclose(f1);
   f1 = fopen("data", "r");
   f2 = fopen("odd", "w");
   f3 = fopen("even", "w");
   while ((num = getw(f1)) != EOF)
   ⚡
     if (num % 2 == 0)
       putw(number, f3);
     else
       putw(number, f2);

```



```

fclose(f1);
fclose(f2);
fclose(f3);
f2 = fopen("odd", "a");
f3 = fopen("even", "w");
printf("odd file\n");
while ((number = getw(f2)) != EOF)
{
    printf("%d", num);
}
printf("even file\n");
while ((number = getw(f3)) != EOF)
{
    printf("%d", num);
}
fclose(f2);
fclose(f3);
}

```

Output

1 2 3 4 5 6 7 8 9 0

odd file

1 3 5 7 9

even file

2 4 6 8 0

1. A file named ODATA contains a series of integers numbers. code a program to read these numbers and then write all odd numbers to a file to be called ODD and all even numbers to a file EVEN.

Ans:-

```
#include <stdio.h>
#include <conio.h>
{
FILE *f1, *f2, *f3;
f1 = fopen("ODATA", "w");
for (i=0; i<35; i++)
{
scanf ("%d", &number);
putw (number, f1);
}
fclose (f1);
f2 = fopen ("ODD", "w");
while (number = getw (f2)) != EOF)
{
if (number % 2 == 0)
putw (number, f4);
else
putw (number, f2)
}
fclose (f2);
f3 = fopen ("EVEN", "w");
f4 = fopen ("ODD", "w");
while (number = getw (f3)) != EOF)
{
printf ("%d\n", number);
fclose (f4);
fclose (f3);
}
```



Output

1 2 3 4 5 6 7 8 9 0

000 file

1 3 5 7 9

EVEN file

2 4 6 8



**SCMS SCHOOL OF ENGINEERING & TECHNOLOGY, VIDYA NAGAR,
KARUKUTTY-683582**

Department of Computer Science & Engineering

Semester 2

CS100 COMPUTER PROGRAMMING
Rubrics

C06

1. What is the purpose of "rb" in fopen() function used below in the code?

```
FILE *fp;  
fp = fopen("source.txt", "rb");
```

- A. open "source.txt" in binary mode for reading
- B. open "source.txt" in binary mode for reading and writing
- C. Create a new file "source.txt" for reading and writing
- D. None of above

2. What does fp point to in the program ?

```
int main()  
{  
    FILE *fp;  
    fp=fopen("trial", "r");  
    return 0;  
}
```

- A. The first character in the file
- B. A structure which contains a char pointer which points to the first character of a file.
- C. The name of the file.
- D. The last character in the file.

3. Which of the following operations can be performed on the file "NOTES.TXT" using the below code?

```
FILE *fp;  
fp = fopen("NOTES.TXT", "r+");
```

- A. Reading
- B. Writing
- C. Appending
- D. Read and Write

4. To print out a and b given below, which of the following printf() statement will you use?


```
float a=3.14;  
double b=3.14;
```

- A. printf("%f %lf", a, b);
- B. printf("%Lf %f", a, b);
- C. printf("%Lf %Lf", a, b);
- D. printf("%f %Lf", a, b);

5. Which files will get closed through the fclose() in the following program?

```
int main()  
{  
    FILE *fs, *ft, *fp;  
    fp = fopen("A.C", "r");  
    fs = fopen("B.C", "r");  
    ft = fopen("C.C", "r");  
    fclose(fp, fs, ft);  
    return 0;  
}
```

- A. "A.C" "B.C" "C.C"
- B. "B.C" "C.C"
- C. "A.C"
- D. Error in fclose()

6. On executing the below program what will be the contents of 'target.txt' file if the source file contains a line "To err is human"?

```
int main()  
{  
    inti, fss;  
    char ch, source[20] = "source.txt", target[20] = "target.txt", t;  
    FILE *fs, *ft;  
    fs = fopen(source, "r");  
    ft = fopen(target, "w");  
    while(1)  
    {  
        ch=getc(fs);  
        if(ch==EOF)  
            break;  
        else  
        {  
            fseek(fs, 4L, SEEK_CUR);  
            fputc(ch, ft);  
        }  
    }  
}
```

```
}  
return 0;  
}
```

- A. r n
- B. Trh
- C. err
- D. None of above

7. Out of fgets() and gets() which function is safe to use?

- A. gets()
- B. fgets()

8. Consider the following program and what will be content of t?

```
#include<stdio.h>  
  
int main()  
{  
    FILE *fp;  
    int t;  
    fp = fopen("DUMMY.C", "w");  
    t = fileno(fp);  
    printf("%d\n", t);  
    return 0;  
}
```

- A. size of "DUMMY.C" file
- B. The handle associated with "DUMMY.C" file
- C. Garbage value
- D. Error in fileno()

9. While calling the fprintf() function in the format string conversion specifier %s can be used to write a character string in capital letters.

- A. True
- B. False

10. A text stream is an ordered sequence of characters composed into lines, each line consisting of zero or more characters plus a terminating new-line character.

- A. True
- B. False

```
}  
return 0;  
}
```

- A. r n
- B. Trh
- C. err
- D. None of above

7. Out of fgets() and gets() which function is safe to use?

- A. gets()
- B. fgets()

8. Consider the following program and what will be content of t?

```
#include<stdio.h>  
  
int main()  
{  
    FILE *fp;  
    int t;  
    fp = fopen("DUMMY.C", "w");  
    t = fileno(fp);  
    printf("%d\n", t);  
    return 0;  
}
```

- A. size of "DUMMY.C" file
- B. The handle associated with "DUMMY.C" file
- C. Garbage value
- D. Error in fileno()

9. While calling the fprintf() function in the format string conversion specifier %s can be used to write a character string in capital letters.

- A. True
- B. False

10. A text stream is an ordered sequence of characters composed into lines, each line consisting of zero or more characters plus a terminating new-line character.

- A. True
- B. False

Answer

1. A

2. B

3. D

4. A

5. D

6. B

7. B

8. B

9. B

10. A

SCMS SCHOOL OF ENGINEERING & TECHNOLOGY, VIDYA NAGAR,
KARUKUTTY-683582
Department of Computer Science & Engineering
Semester 2
CS100 COMPUTER PROGRAMMING
Rubrics

20/20
[Handwritten signature]

1. What is the purpose of "rb" in fopen() function used below in the code?

```
FILE *fp;  
fp = fopen("source.txt", "rb");
```

- A. open "source.txt" in binary mode for reading
- B. open "source.txt" in binary mode for reading and writing
- C. Create a new file "source.txt" for reading and writing
- D. None of above

2. What does fp point to in the program ?

```
int main()  
{  
    FILE *fp;  
    fp=fopen("trial", "r");  
    return 0;  
}
```

- A. The first character in the file
- B. A structure which contains a char pointer which points to the first character of a file.
- C. The name of the file.
- D. The last character in the file.

3. Which of the following operations can be performed on the file "NOTES.TXT" using the below code?

```
FILE *fp;  
fp = fopen("NOTES.TXT", "r+");
```

- A. Reading
- B. Writing
- C. Appending

D. Read and Write

4. To print out a and b given below, which of the following printf() statement will you use?

```
float a=3.14;  
double b=3.14;
```

- A. printf("%f %lf", a, b);
- B. printf("%Lf %f", a, b);
- C. printf("%Lf %Lf", a, b);
- D. printf("%f %Lf", a, b);

5. Which files will get closed through the fclose() in the following program?

```
int main()  
{  
    FILE *fs, *ft, *fp;  
    fp = fopen("A.C", "r");  
    fs = fopen("B.C", "r");  
    ft = fopen("C.C", "r");  
    fclose(fp, fs, ft);  
    return 0;  
}
```

- A. "A.C" "B.C" "C.C"
- B. "B.C" "C.C"
- C. "A.C"
- D. Error in fclose()

6. On executing the below program what will be the contents of 'target.txt' file if the source file contains a line "To err is human"?

```
int main()  
{  
    inti, fss;  
    char ch, source[20] = "source.txt", target[20] = "target.txt", t;  
    FILE *fs, *ft;  
    fs = fopen(source, "r");  
    ft = fopen(target, "w");  
    while(1)  
    {  
        ch = getc(fs);  
        if(ch == EOF)  
            break;  
        else  
        {  
            fseek(fs, 4L, SEEK_CUR);  
            t = ch;  
            fputc(t, ft);  
        }  
    }  
}
```



```

fputc(ch, ft);
    }
}
return 0;
}

```

- A. r n
- B. Trh
- C. err
- D. None of above

7. Out of fgets() and gets() which function is safe to use?

- A. gets()
- B. fgets()

8. Consider the following program and what will be content of t?

```

#include<stdio.h>

int main()
{
    FILE *fp;
    int t;
    fp = fopen("DUMMY.C", "w");
    t = fileno(fp);
    printf("%d\n", t);
    return 0;
}

```

- A. size of "DUMMY.C" file
- B. The handle associated with "DUMMY.C" file
- C. Garbage value
- D. Error in fileno()

9. While calling the fprintf() function in the format string conversion specifier %s can be used to write a character string in capital letters.

- A. True
- B. False

10. A text stream is an ordered sequence of characters composed into lines, each line consisting of zero or more characters plus a terminating new-line character.

- A. True
- B. False

SCMS SCHOOL OF ENGINEERING & TECHNOLOGY, VIDYANAGAR,
KARUKUTTY-683582

Department of Computer Science & Engineering

Semester 2

CS100 COMPUTER PROGRAMMING

Rubrics

16/20

1. What is the purpose of "rb" in fopen() function used below in the code?

```
FILE *fp;  
fp = fopen("source.txt", "rb");
```

- A. open "source.txt" in binary mode for reading
- B. open "source.txt" in binary mode for reading and writing
- C. Create a new file "source.txt" for reading and writing
- D. None of above

2. What does fp point to in the program ?

```
int main()  
{  
    FILE *fp;  
    fp=fopen("trial", "r");  
    return 0;  
}
```

- A. The first character in the file
- B. A structure which contains a char pointer which points to the first character of a file.
- C. The name of the file.
- D. The last character in the file.

3. Which of the following operations can be performed on the file "NOTES.TXT" using the below code?

```
FILE *fp;  
fp = fopen("NOTES.TXT", "r+");
```

- A. Reading
- B. Writing
- C. Appending

D. Read and Write

4. To print out a and b given below, which of the following printf() statement will you use?

```
float a=3.14;  
double b=3.14;
```

- A. printf("%f %lf", a, b);
- B. printf("%Lf %f", a, b);
- C. printf("%Lf %Lf", a, b);
- D. printf("%f %Lf", a, b);

5. Which files will get closed through the fclose() in the following program?

```
int main()  
{  
    FILE *fs, *ft, *fp;  
    fp = fopen("A.C", "r");  
    fs = fopen("B.C", "r");  
    ft = fopen("C.C", "r");  
    fclose(fp, fs, ft);  
    return 0;  
}
```

- A. "A.C" "B.C" "C.C"
- B. "B.C" "C.C"
- C. "A.C"
- D. Error in fclose()

6. On executing the below program what will be the contents of 'target.txt' file if the source file contains a line "To err is human"?

```
int main()  
{  
    inti, fss;  
    char ch, source[20] = "source.txt", target[20] = "target.txt", t;  
    FILE *fs, *ft;  
    fs = fopen(source, "r");  
    ft = fopen(target, "w");  
    while(1)  
    {  
        ch=getc(fs);  
        if(ch==EOF)  
            break;  
        else  
        {  
            fseek(fs, 4L, SEEK_CUR);  
        }  
    }  
}
```



```
fputc(ch, ft);
}
}
return 0;
}
```

IND

- A. r n
- B. Trh
- C. err
- D. None of above

Imj

C1

1.

2.

3

7. Out of fgets() and gets() which function is safe to use?

- A. gets()
- B. fgets()

8. Consider the following program and what will be content of t?

```
#include<stdio.h>

int main()
{
    FILE *fp;
    int t;
    fp = fopen("DUMMY.C", "w");
    t = fileno(fp);
    printf("%d\n", t);
    return 0;
}
```

- A. size of "DUMMY.C" file
- B. The handle associated with "DUMMY.C" file
- C. Garbage value
- D. Error in fileno()

9. While calling the fprintf() function in the format string conversion specifier %s can be used to write a character string in capital letters.

- A. True
- B. False

10. A text stream is an ordered sequence of characters composed into lines, each line consisting of zero or more characters plus a terminating new-line character.

- A. True
- B. False

INDUSTRIAL RELEVANCE OF C PROGRAMMING LANGUAGE

Importance of 'C' language

C language is a famous programming language due to its qualities. Some qualities are:

1. It is robust language whose rich setup of built in functions and operator can be used to write any complex program.
2. Program written in C are efficient due to several variety of data types and powerful operators.
3. The C compiler combines the capabilities of an assembly language with the feature of high level language. Therefore it is well suited for writing both system software and business package.
4. There are only 32 keywords; several standard functions are available which can be used for developing program.
5. C is portable language; this means that C programs written for one computer system can be run on another system, with little or no modification.
6. C language is well suited for structured programming, this requires user to think of a problems in terms of function or modules or block. A collection of these modules make a program debugging and testing easier.
7. C language has its ability to extend itself. A c program is basically a collection of functions that are supported by the C library. We can continuously add our own functions to the library with the availability of the large number of functions. In India and abroad mostly people use C programming language because it is easy to learn and understand.

The reasons to use C for the following:

1. **C is one of the foundations for modern information technology (IT) and computer science (CS).**
Many working principles of IT and CS, such as programming languages, computer architectures, operating systems, network communication, database, graphical user interface (GUI), graphics, image processing, parallel processing, multi-threads, real-time systems, device drivers, data acquisition, algorithms, numerical analysis, and computer game, are based on or reflected in the functionalities and features of C. The experience in C will help students understand the working principles of these important concepts in IT and CS. Therefore, C is required for the CS major in almost all universities.
2. **C is the most commonly used programming language in industry.**
Academic institutions have a mission to teach technologies that are widely used in the real world so that students have the skills and knowledge that employers need. More than 90 percent of the programs running on our desktops, from operating systems and e-mail clients to Web browsers and word processors, are written in C or its relative, C++ which has extensions to C. Most games and underlying robot control software are written in C or C++. With the

knowledge of C, students will not only be able to play games and robots, but also understand their underlying working principles and potentially develop their own games and robots.

3. **C is the language of choice for programming embedded and mechatronic systems with hardware interfaces.**
4. **C is one of the most commonly used programming languages in colleges and universities.**
Computer programming is an essential skill for advanced studies in Science, Technology, Engineering, and Mathematics (STEM) fields. Like in industry, C is also one of the most commonly used programming languages in colleges and universities for teaching and research.
5. **C is the base for almost all popular programming languages.**
C is the language of choice for system programming. Because of the performance and portability of C, almost all popular cross-platform programming languages and scripting languages, such as C++, Java, Python, Objective-C, Perl, Ruby, PHP, Lua, and Bash, are implemented in C and borrowed syntaxes and functions heavily from C. They share the similar operators, expressions, repetition statements, control structures, arrays, input and output, and functions.
6. **C excels as a model of programming languages.**
C does an excellent job of illustrating the underlying working principles of computers, scientific computing, and disciplined software development. Students gain valuable knowledge of such fundamental programming concepts as data types, internal data representations, operators, expressions, loops for repetitions, control structures, arrays, input and output, functions, debugging, etc. Studying C provides a solid foundation for students who want to learn advanced programming skills such as object-oriented programming, event-driven programming, multi-thread programming, real-time programming, embedded programming, network programming, parallel programming, other programming languages, and new and emerging computing paradigms such as grid-computing and cloud computing.
7. **Once students have learned C, they can pick up any other languages by themselves.**
Certain languages and tools are typically used to solve domain specific problems. Therefore, the ability to understand and learn new languages is important. All other modern languages borrowed heavily from C. Once students learned C, it is easy for them to learn by themselves any other computer languages without much difficulty.
8. **C is a standardized programming language with international standards.**
A standardized programming language is stable and its evolution is overseen by a technical standard committee made up of business, academic, and organizational representatives with a stake in the language.
9. **Computer programming is becoming a necessary skill for many professions.**
Writing computer programs is essential to solving complex science and engineering problems. Many principles and concepts in STEM disciplines can be illustrated and reinforced through writing programs. C and C++ are more widely used in STEM fields than any other programming languages such as Java, Fortran, or Matlab.
10. **Computer programming can develop student's critical thinking capabilities.**
Developing a program to solve a practical problem involves many creative works, including design, logic reasoning, math, etc. It can help students find practical applications of many math concepts such as variables in Algebra I and trigonometry. Debugging a program can

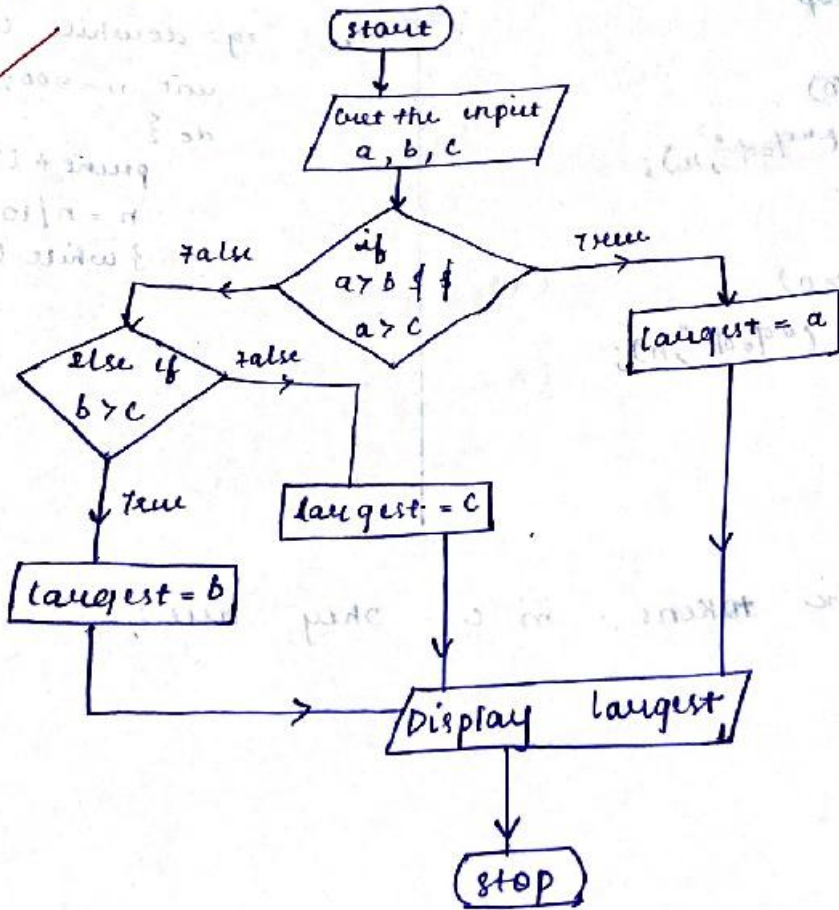
also help student improve their reasoning and logical thinking capabilities. The computer-aided problem solving capabilities can be trained using C.



48

```
#include <stdio.h>
#include <conio.h>
void main ()
{
    int a,b,c ,largest ;
    clrscr();
    printf ("Enter the three numbers ");
    scanf ("%d %d %d", &a, &b, &c);
    if
    largest = (a > b && a > c) ? a : ((b > c) ? b : c);
    printf ("largest is %d", largest);
    getch ();
}
```

Flowchart



2. Output of the following code: 0011

```
int i=-1, j=-1, k=0, l=2, m;
m = i++ || j++ || k++ || l--1;
printf("%d %d %d %d %d", i, j, k, l, m);
```

In ②, the values of i, j and k are assigned to m and then its value is incremented since it is a postfix.

∴ $m = -1 || -1 || 0 || 1$ is the expression. The overall expression yields the value 1.

∴ output becomes ~~0,0,1~~ 0011

3. Entry controlled loop

- Condition is evaluated at the beginning of the loop. i.e. before entering into the loop.
- Loop will only get executed if when the condition gets satisfied.

eg: while loop

```
int n=50;
while (n > 0)
{
    printf("%d", n);
    n--;
}
```

```
while (n > 0)
{
    printf("%d", n);
    n--;
}
```

exit controlled loop
condition is evaluated at the end of the loop.

loop gets executed even if the condition is false. evaluates before the condition.

eg: do while loop

```
int n=500;
do
{
    printf("%d", n);
    n = n/10;
} while (n > 0);
```

4. There are six tokens in C. They are:

- keywords
- identifiers
- constants
- special symbols
- white spaces
- ~~variational~~ logical operators.

Keywords → There are 32 keywords in C. These are words defined in the library.

Identifiers → Every word in C is either an identifier or a keyword.

constants → Their value doesn't change. There are 4 types
- Integer constant
- character constant.

Special symbols → Special symbols include #, %, &, \$, /, \, !, ^, ", ', * etc.

white space → \n — newline
\r — carriage return
\t — horizontal tab space etc.

operators include arithmetic and logical operators.

arithmetic operators → +, -, /, *

logical operators → >, <, >=, <=, !=, == etc.

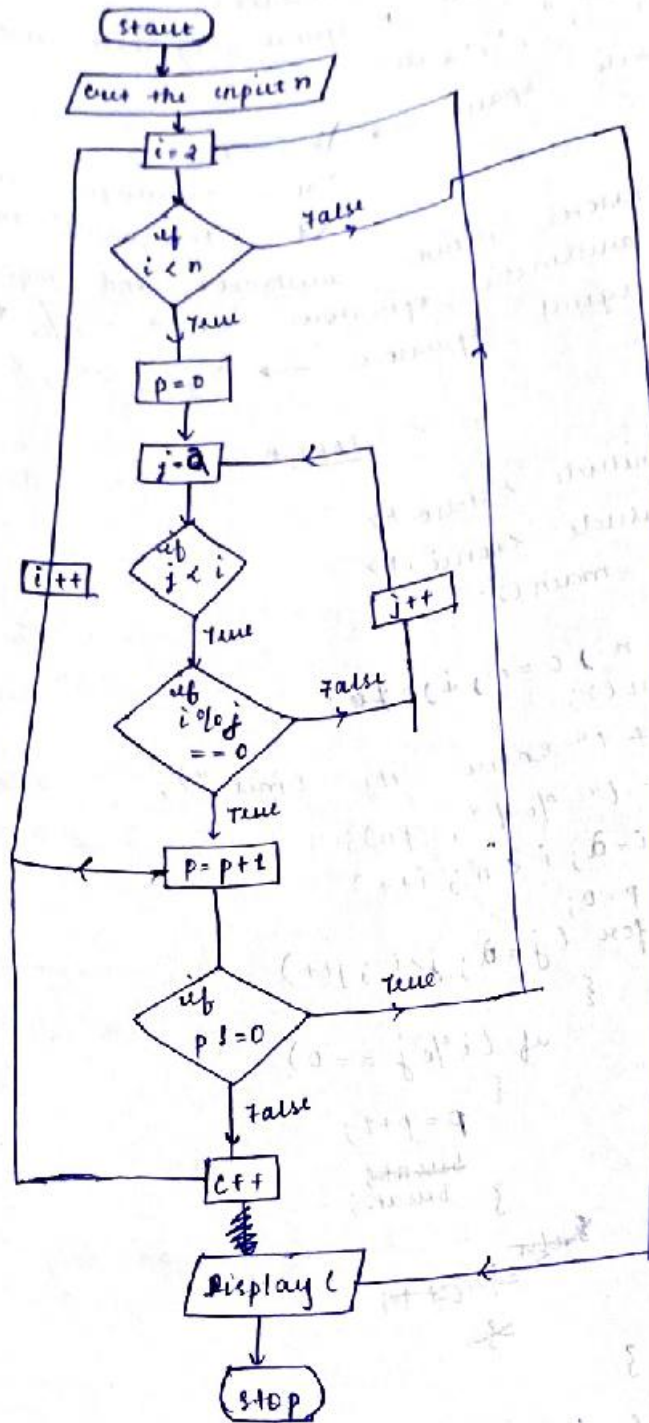
PART B

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int n, c = 0, i, p = 0;
    clrscr();
    printf("Enter the limit");
    scanf("%d", &n);
    for (i = 2; i < n; i++)
    {
        p = 0;
        for (j = 2; j < i; j++)
        {
            if (i % j == 0)
            {
                p = p + 1;
                break;
            }
        }
        if (p == 0)
        {
            printf("%d ", i);
            c++;
        }
    }
    printf("There are %d prime numbers in the limit %d, c, n);", c, n);
}
```

scanf (

```
getch ();  
}
```

flowchart



```
2. # include <stdio.h>
# include <conio.h>
void main ()
{ int n, m; a=0, b=1 b=1, x) fact=1) i;
  clrscr ();
  printf (" 1. factorial 2. fibonacci ");
  printf ("\n enter the choice ");
```



SCMS SCHOOL OF ENGINEERING & TECHNOLOGY

2

```

scanf ("%d", &x);
if (x == 1)
    printf ("enter the number");
    printf ("");
    scanf ("%d", &n);
    for (i=1; i <= n; i++)
        fact = fact * i;
    printf ("Factorial of %d is %d", n, fact);
}
else
    printf ("%d %d", a, b);
    printf ("enter the limit");
    scanf ("%d", &m);
    for (i=0; i < m; i++)
        {
            c = a+b;
            a = b;
            b = c;
            printf ("%d %d", a, b);
            printf ("%d | c);
        }
    }

```

0 1 | 2

printf ("a, b")

switch ?

```

getch ();
}

```

```

3(a) #include <stdio.h>
#include <conio.h>
void main()
{
    int a, n, rev, num = 0;
    clrscr ();
    printf ("enter the number ");
    scanf ("%d", &n);
    a = n;
    while (n > 0)
    {
        rev = n % 10;
        n = n / 10;
        num = num * 10 + rev;
    }
}

```

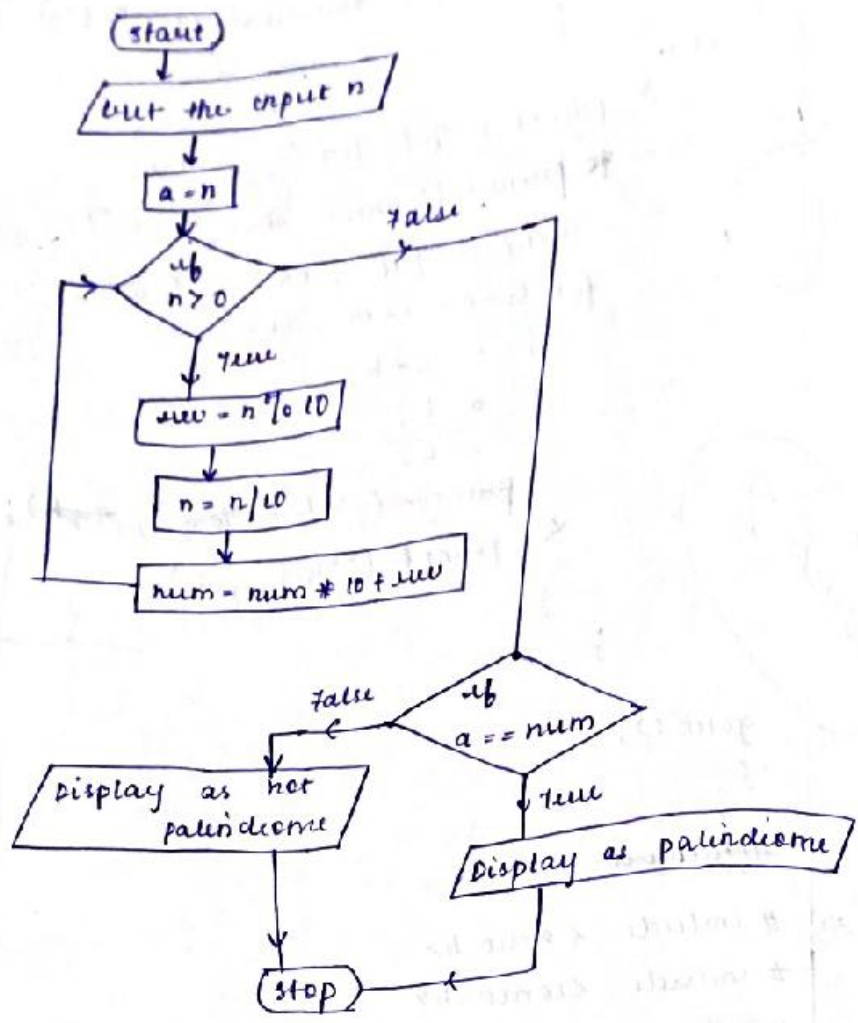


```

if (a == num)
{
    printf ("the given number %d is palindromic\n", n);
}
else
{
    printf ("the given number %d is not palindromic\n", n);
}
getch();
}

```

Flowchart



(b) Break statement is used to break out from a

```

eg. if (n > 0)
{
    p++;
    break;
}
else
{
    p = 1;
}
statement x;

```

on reaching the break statement, it exits from loop and jumps to the statement after the loop



SCMS SCHOOL OF ENGINEERING & TECHNOLOGY

Continue statement is used to skip the particular iteration and go continue with the next iteration (in the same loop).

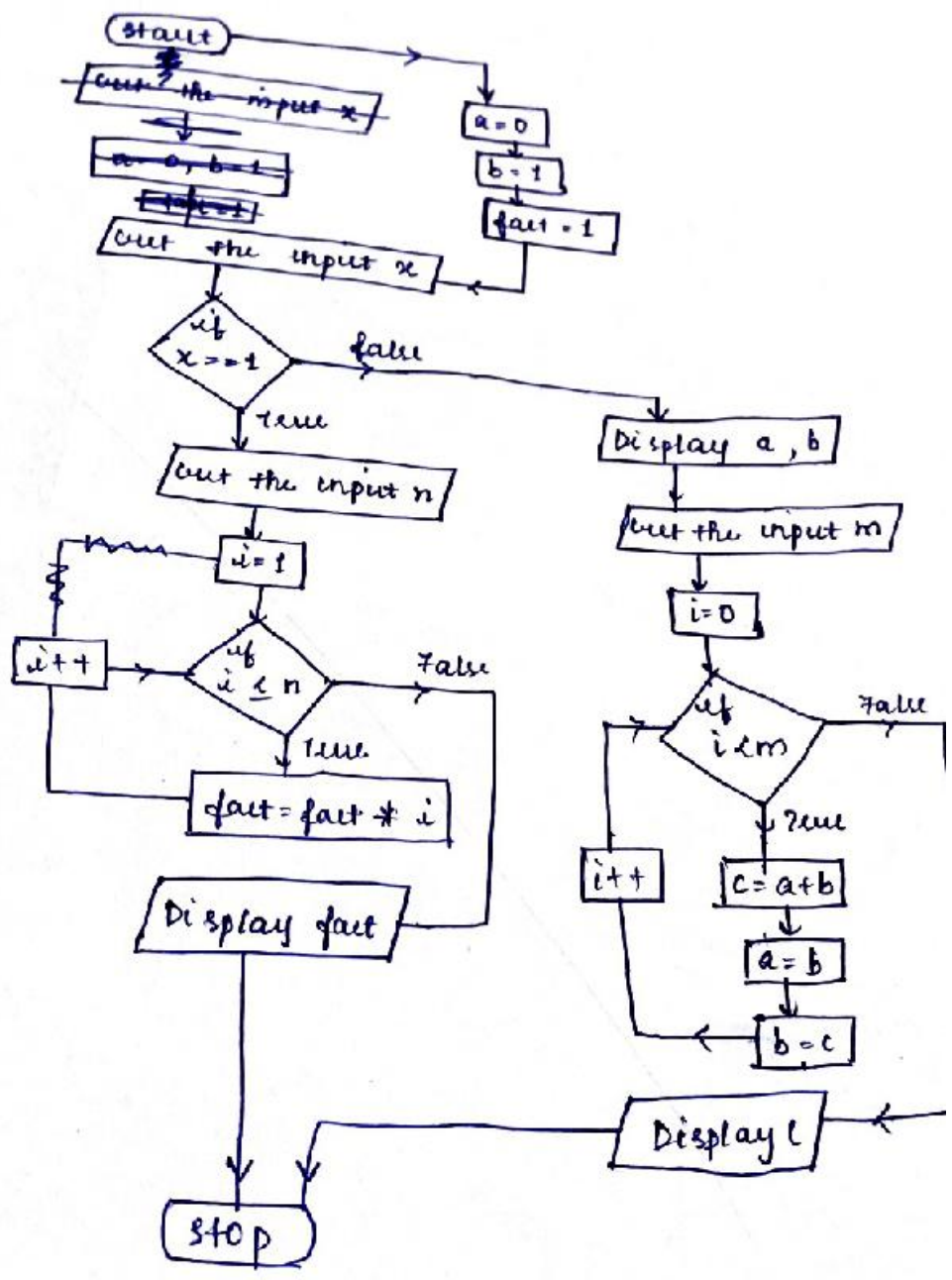
```

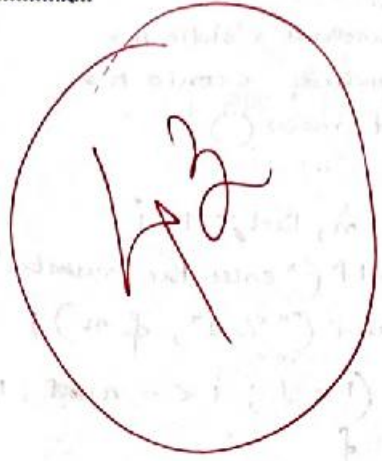
eg: for (i=0; i<n; i++)
{
  if (i==2)
  {
    continue;
  }
  else
  {
    printf("%d", i);
  }
}

```

on reaching the continue statement, that particular iteration is skipped by the compiler and it continues with the next iteration.

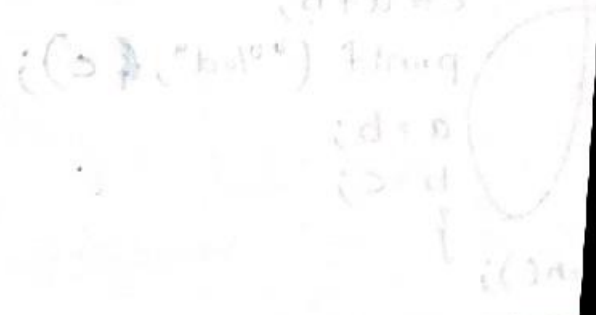
Flowchart





1, PART-B

```
#include <stdio.h>
#include <conio.h>
void main ()
{
  int l, h, count, i, j, c, count=0;
  clrscr();
  printf("enter lower limit");
  scanf ("%d", &l);
  printf ("enter upper limit");
  scanf ("%d", &h);
  for (i=l; i<=h; i++)
  {
    c = 0;
    for (j=2; j<=i; j++)
    {
      if (i%j == 0)
      {
        c = 1;
        break;
      }
      else
      {
        // count = count + 1;
      }
    }
    if (c == 0)
    {
      count = count + 1;
    }
  }
  printf ("count is %d", count);
}
```




```

getch();
}
(i)
#include <stdio.h>
#include <conio.h>
void main()
{
int n, fact = 1, i;
printf("enter the number");
scanf("%d", &n);
for (i = 1; i <= n; i++)
{
fact = fact * i;
}
printf("factorial is %d", fact);
}

```

```

factorial
n = int(input)
fact = 1
for i in range(1, n+1):
    fact = fact * i
print(fact)

```

```

n =
a = 0
b = 1
print a
print b
for

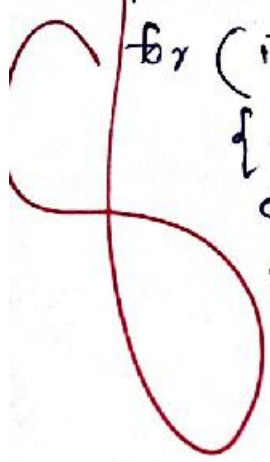
```

```

(ii)
#include <stdio.h>
#include <conio.h>
void main()
{
int n, a, b, i, c;
clrscr();
printf("enter the limit");
scanf("%d", &n);
a = 0;
b = 1;
printf("%d", a);
printf("%d", b);
for (i = 0; i <= n i <= n; i++)
{
c = a + b;
printf("%d", c);
a = b;
b = c;
}
getch();
}

```

Menu driven



```

#include <stdio.h>
#include <conio.h>
void main ()
{
    int n, a, r, sum = 0;
    printf("enter the number");
    scanf("%d", &n);
    a = n;
    while (n > 0)
    {
        r = n % 10;
        sum = (sum * 10) + r;
        n = n / 10;
    }
    if (sum == a)
    {
        printf("no. is palindrome");
    }
    else
    {
        printf("no. is not palindrome");
    }
    getch();
}

```

```

a = n
sum = 0
n = int(input(no))
while (n > 0)
    r = n % 10
    sum = (sum * 10) + r
    n = n // 10
if (sum == a):
    pal
else:
    not pal

```

Break statement is used to exit the loop and to statements out of the loop. Break is used to exit the loop and to execute the rest out of the loop.

eg:

switch (cond)

{

case value 1 :
exp 1
break

case value 2 :
exp 2
break

~~case~~
}

19/11/21

void
()
x, y
for ()
int ()
int
int

These switch will check the condition and reads the case values, if the case value is 1 exp 1 gets executed and thus it breaks the loop and execute the statements

- Continue is used to skip the rest of the statements a loop and to start with the loop again. Continue statement can't exit the loop as in the case of break

```
for (i=0; i<n; i++)  
{  
  if (cond)  
  {  
    statement-1  
    statement-2  
    continue  
    statement-3  
    statement-4  
  }  
}
```

Here the compiler will check the condition and if it is true statement-1 gets executed, then statement-2 if it is true

```
for (i=0; i<n; i++)  
{  
  if (cond)  
  {  
    statement-1  
    continue  
    statement-2  
    statement-3  
  }  
}
```

Here when the loop starts, compiler will check the condition and if it is true statement-1 gets executed and it will again go back to the loop by skipping the rest of the statements i.e., statement-2 and statement-3 until the condition fails. Hence exiting from the loop is not possible for continue

While 'goto' statement can exit from the entire loop and go to some other statements.



PART-A

- 3) - Entry controlled loop will check the condition at first and then the body of loop gets executed. If the condition given in the statement is true, body of loop gets executed, otherwise execution will not take place.

eg: for loop, while loop.

```
while (cond)
{
    body of loop
}
```

```
while (n > 0)
```

```
{
    r = n % 10;
    sum = (sum * 10) + r;
    n = n / 10;
}
```

Here, first of all the condition ($n > 0$) is evaluated. If the condition is true, then the body of the loop gets executed; otherwise body of loop will not be executed.

Exit control loop will check the condition only after executing the body of the loop once. After the execution of loop one-time, then the condition is checked and if it is true, the body of loop will be executed again.

eg: do-while.

```
do
{
```

```
do
{
```

```
printf("enter a, b, c");
```

```
scanf("%d %d %d", &a, &b, &c);
```

```

if (a > b && a > c)
{
printf ("a is greater");
}
else if (b > c)
printf ("b is greater");
else
printf ("c is greater");
printf ("press y if you want to continue else n");
scanf ("%c", &n);
while (n == 'y')

```

- 4, Tokens are the smallest components in C. It includes identifier, operator, ~~value~~, special characters, constants, data value.
- Identifier is the name assigned to the data value.
 - Operator is used to do operations such as addition, subtraction, assignment etc.
 - Special characters such as ~~/*~~ ; , etc are in C.
 - Constants are the value assigned to the variable. It can't be changed while variable's value can be changed accordingly.

Data type refers to the type of the data we are providing such as int, float, double etc.



```

2) int i=-1, j=-1, k=0, l=1, m;
   m=i++ && j++ && k++ || --l;
   printf ("%d %d %d %d %d", i, j, k, l, m);

```

Output will be:
 0 0 1 1 1

$i++$ will give the output $\leftarrow 0$ i.e., $-1++1$ which gives 0.

$j++$ will give 0, $-1++1$ which is 0.

$k++$ will give 1, $0++1$ which is 1

$--l$ will give 1, $--2$ which is 1.

By using logical operators from left to right. 0 gives 0, $0 \&\& 1$ will give 0 and $0 \parallel 1$ will give 1.

```

#include <stdio.h>
#include <conio.h>
void main()

```

```

{
  int a, b, c;

```

```

  printf ("enter a, b, c");
  scanf ("%d %d %d", &a, &b, &c);

```

```

  (a > b && a > c) ? printf ("a is greater") : (b > c ? printf ("b is greater") : printf ("c is greater"));

```

```

  getch();

```

```

}

```




SCMS SCHOOL OF ENGINEERING & TECHNOLOGY

Subject: Computer Programming

Part - A

20/2

```

1. #include <conio.h>
   #include <stdio.h>
   void main()
   {
       int x, y, z, largest;
       clrscr();
       printf("Enter 3 nos: ");
       scanf("%d %d %d", &x, &y, &z);
       // largest = x > y ? (x > z ? x : z) : (y > z ? y : z);
       printf("largest no. is: %d", largest);
       getch();
   }

```

```

if (a > b) & (a > c)
    largest = a;
else if (b > c)
    largest = b;
else
    largest = c;

```

24/2

3. Entry controlled loop

Eg: for-loop, while loop

Here, the body inside the curly bracket, or the body of is executed only if the condition given at the start is true.

The entry to the body of the loop is controlled by a condition before it's loop; hence the name entry controlled loop.

```

while (a > b)
{
    Body of the loop
}

```



Exit controlled loop

do-while loop

Here, the body of the loop gets executed once before the condition itself. i.e., even if the condition is false, the body is executed at least once.

get executed once. This means that the exit of the loop is controlled by a condition statement, whereas the entry is

```
eg: do  
    { Body of do-while loop  
    :  
    } while (a > b),
```

4. Tokens

Tokens are the fundamental building blocks of a C program. They are the smallest indivisible parts comprising the program. The tokens in C are:

1) Keywords

They are irreplaceable words that convey special meaning to the compiler. There are no other words to replace these in C.

Eg: int, float, main, char etc.

2) Identifiers

They are used by the compiler to identify various data. Eg: int xtreme, char name etc.

There are some rules regarding the naming of identifiers.

- It should not begin with a number.
- White spaces are not allowed.
- Only special character allowed is "_".
- Only the first 31 characters are significant.

Constants

These are used to identify datatypes that never change their value in the program. This means that they have a fixed value.

```
const int S;
```

i) Variables

These are used to identify data types that will change their values or can change their values.

Eg: int x = 5;

ii) Strings

These are a group of data of similar datatype referenced under a single name.

Eg: char x[10];

PART-B

1. #include <conio.h>

#include <stdio.h>

void main()

{ int n, ~~count~~, i, j, flag=0, count=2;

clrscr();

printf("Enter the limit : ");

scanf("%d", &n);

for (i=0; i<n; i++)

{ flag=0;

for (j=2; j<n; j++)

{ if (i%j==0)

{ flag=1;

break;

}

}

if (flag==0)

{ count++;

}

printf("Prime numbers in the limit: %d", count);

getch();

}

WJ

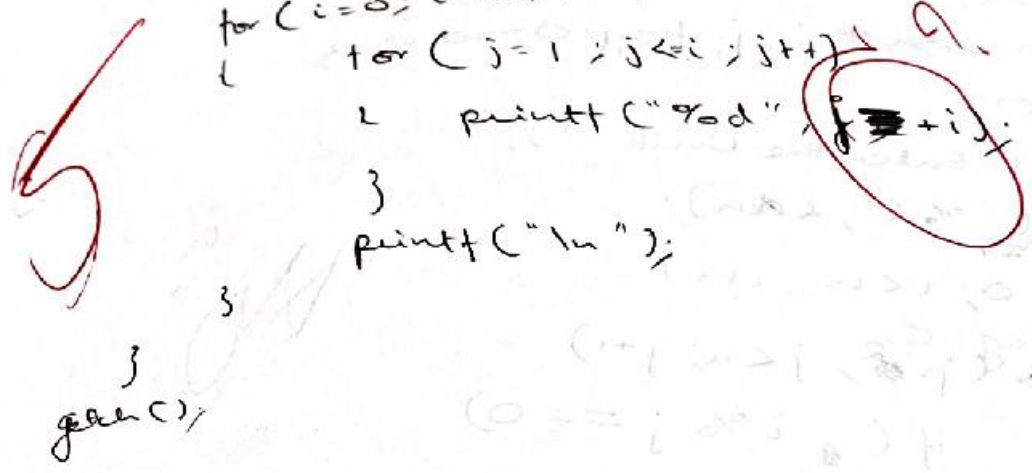
~~if (i%j==0)~~

(14) ...

9. Array Note

```
#include <stdio.h>
#include <math.h>
void main()
{
    int choice, n, i, fact = 1;
    clrscr();
    printf("Enter n: ");
    scanf("%d", &n);
    if (choice == 1)
    {
        printf("Enter the number: ");
        scanf("%d", &n);
        for (i = 1; i <= n; i++)
        {
            fact = fact * i;
        }
        printf("Factorial of %d is %d", n, fact);
    }
}
```

```
3
clr
{
    printf("Enter the limit: ");
    scanf("%d", &n);
    for (i = 1; i <= n; i++)
    {
        for (j = 1; j <= i; j++)
        {
            printf("%d ", i * j);
        }
        printf("\n");
    }
}
```



```
#include <conio.h>
#include <math.h>
#include <stdio.h>
```

```
void main()
```

```
{
    clrscr();
    int n, i, r, sum = 0;
    clrscr();
```

```
printf("Enter the number: ")
```

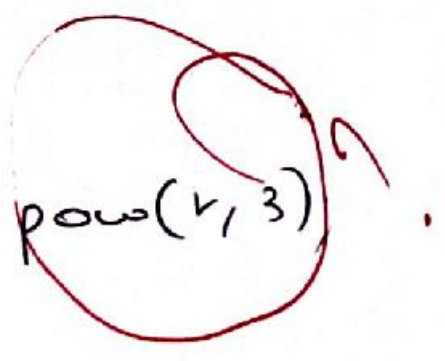
```
scanf("%d", &n);
a = 0;
while (a > 0)
```

```
{
    r = a % 10;
```

```
sum = sum + pow(r, 3);
```

```
a = a / 10;
```

```
}
```





```
if (num == n)
    printf("Palindrome");
else
    printf("Not Palindrome");
getch();
}
```

3

6) Break

It is used by the user to make the compiler understand that if this command is encountered, the flow of control is taken outside the first loop above it.

Eg: switch (ch)

```
{
    case '1': printf("Hi");
             break;
    case '2': printf("Hello");
             break;
}
```

If the value of ch is 1, the output will be "Hi". Just as break statement is encountered, the flow of ^{control} ~~code~~ comes out of the switch body & reaches the next statement without going the case '2' of the loop.

Continue

Continue is generally used to force the compiler to skip next iteration of the loop above it without executing commands under it.

```
Eg: for (int i=0; i<n; i++)
    {
        if (i==3)
            continue;
    }
```

Here, when the i val if condition is satisfied, continue statement directly, the next iteration i = 4 is executed under

PART - A

1) // to find largest of 3

```
#include <stdio.h>
```

```
#include <conio.h>
```

```
void main ()
```

```
{
```

```
int a, b, c, l;
```

```
clrscr();
```

```
printf("enter a, b, c);
```

```
scanf("%d%d%d", &a, &b, &c);
```

```
if (a > b && a > c)
```

```
{
```

```
l l = a;
```

```
}
```

```
else
```

```
{
```

```
if (c > b)
```

```
l = c;
```

```
else
```

```
l = b;
```

```
}
```

```
printf("%d is largest", l);
```

```
getch();
```

```
}
```

24/2

2) Output :

1 1 2 0 0

3) Entry control loop : first the condition variable is evaluated and if ~~the~~ condition is satisfied, then only the body of loop gets executed. For eg:

```
for (int i=0; i<4; i++)
```

```
{  
    printf ("%d", i);  
}
```

gives the output : 0 1 2 3

Exit control loop: In this case, the body of the loop is executed once before evaluation of condition. So the body is executed one extra time. For example,

```
i = 0;  
do {
```

```
    printf ("%d", i);
```

```
    i++;  
} while (i < 4);
```

~~gives output~~ : The condition $i < 4$ is checked after displaying i once, unlike in for loop.

PART-B

1) // No. of prime no. in limit

#include <stdio.h>

#include <conio.h>

void main()

{

int l, u, i, c

printf("lower & upper limits : ");

scanf("%d %d", &l, &u);

for(~~i=0~~, i=l; i<=u; i++)

{

for(j=2; j<i; j++)

{

if(i%j==0)

break;

}

else

c++;

~~c++;~~

}

printf("There are %d prime numbers", c);

getch();

}

NO?

3) // factorial of n & fibonacci in limits

```
#include <stdio.h>  
#include <conio.h>
```

```
void main()  
{
```

```
int ch, n, f, x, a, b, c = 0;
```

```
clrscr();
```

```
printf("1. factorial of a number\n");
```

```
printf("2. fibonacci till limit");
```

```
scanf("%d", &ch);
```

```
switch (ch)  
{
```

```
case 1:
```

```
printf("enter n");
```

```
scanf("%d", &n);
```

```
f = 1;  
while (n > 0)  
{
```

```
    f = f * n;
```

```
    n--;
```

```
}
```

```
printf("factorial is %d", f);  
break;
```




90
11
12
13

Case 2:

```

printf("enter the limit: ");
scanf("%d", &n);
a = 0;
b = 1;
while (c < n)
{
    printf("%d\n%d\n", a, b);
    while (c < n)
    {
        fib = a + b;
        a = b;
        b = fib;
        printf("%d\n", fib);
        c++;
    }
    break;
}
getch();
}

```



}
getch();
}

gives output 1 2 3 4 5
when i became 5, break statement would
be executed and the values 5, 7, 9 were not
displayed.

```
// palindrome check
#include <stdio.h>
#include <conio.h>
void main()
```

```
{
    int a a, s = 0, n;
    clrscr();
    printf("enter number number");
    scanf("%d", &n);
```

```
    a = n;
```

```
    while (n > 0)
    {
        s = (s * 10) + (n % 10);
        n = n / 10;
    }
```

```
    if (s == a)
```

```
        printf("palindrome");
```

```
    else
```

```
        printf("not a palindrome.");
```

```
    getch();
```

```
}
```



Break and Continue statements

break: This statement is used to terminate a loop or switch statement and transfer control outside the loop or switch.

Upon encountering break statement, execution of loop or switch's body will stop at that point and control goes to the next statement in main() function. Example:

```
void main ()
{
    int i = 1
    while (i < 10)
    {
        printf ("%d", i);
        i++;
        if (i == 5)
            break;
    }
}
```

gives output : 1 2 3 4

when i became 5, break statement caused termination of loop and values 5, 6, 7, 8, 9 were not displayed.

ii) Continue : This statement may be used in a loop for skipping execution of body of loop for a particular value of the condition variable. Unlike in the break statement, using continue in a loop will cause it to leave out only the values specified, while ~~to~~ executing the rest. Example,

```

void main ()
{
    int i = 1;
    while (i < 10)
    {
        printf("%d", i);
        i++;
        if (i == 5)
            continue;
    }
}

```

gives the output : 1 2 3 4 6 7 8 9

Here it was not displayed only when its value became 5.

5 1 : try this comp
 because translate should 2 error; make
 for error 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 250, 251, 252, 253, 254, 255, 256, 257, 258, 259, 260, 261, 262, 263, 264, 265, 266, 267, 268, 269, 270, 271, 272, 273, 274, 275, 276, 277, 278, 279, 280, 281, 282, 283, 284, 285, 286, 287, 288, 289, 290, 291, 292, 293, 294, 295, 296, 297, 298, 299, 300, 301, 302, 303, 304, 305, 306, 307, 308, 309, 310, 311, 312, 313, 314, 315, 316, 317, 318, 319, 320, 321, 322, 323, 324, 325, 326, 327, 328, 329, 330, 331, 332, 333, 334, 335, 336, 337, 338, 339, 340, 341, 342, 343, 344, 345, 346, 347, 348, 349, 350, 351, 352, 353, 354, 355, 356, 357, 358, 359, 360, 361, 362, 363, 364, 365, 366, 367, 368, 369, 370, 371, 372, 373, 374, 375, 376, 377, 378, 379, 380, 381, 382, 383, 384, 385, 386, 387, 388, 389, 390, 391, 392, 393, 394, 395, 396, 397, 398, 399, 400, 401, 402, 403, 404, 405, 406, 407, 408, 409, 410, 411, 412, 413, 414, 415, 416, 417, 418, 419, 420, 421, 422, 423, 424, 425, 426, 427, 428, 429, 430, 431, 432, 433, 434, 435, 436, 437, 438, 439, 440, 441, 442, 443, 444, 445, 446, 447, 448, 449, 450, 451, 452, 453, 454, 455, 456, 457, 458, 459, 460, 461, 462, 463, 464, 465, 466, 467, 468, 469, 470, 471, 472, 473, 474, 475, 476, 477, 478, 479, 480, 481, 482, 483, 484, 485, 486, 487, 488, 489, 490, 491, 492, 493, 494, 495, 496, 497, 498, 499, 500, 501, 502, 503, 504, 505, 506, 507, 508, 509, 510, 511, 512, 513, 514, 515, 516, 517, 518, 519, 520, 521, 522, 523, 524, 525, 526, 527, 528, 529, 530, 531, 532, 533, 534, 535, 536, 537, 538, 539, 540, 541, 542, 543, 544, 545, 546, 547, 548, 549, 550, 551, 552, 553, 554, 555, 556, 557, 558, 559, 560, 561, 562, 563, 564, 565, 566, 567, 568, 569, 570, 571, 572, 573, 574, 575, 576, 577, 578, 579, 580, 581, 582, 583, 584, 585, 586, 587, 588, 589, 590, 591, 592, 593, 594, 595, 596, 597, 598, 599, 600, 601, 602, 603, 604, 605, 606, 607, 608, 609, 610, 611, 612, 613, 614, 615, 616, 617, 618, 619, 620, 621, 622, 623, 624, 625, 626, 627, 628, 629, 630, 631, 632, 633, 634, 635, 636, 637, 638, 639, 640, 641, 642, 643, 644, 645, 646, 647, 648, 649, 650, 651, 652, 653, 654, 655, 656, 657, 658, 659, 660, 661, 662, 663, 664, 665, 666, 667, 668, 669, 670, 671, 672, 673, 674, 675, 676, 677, 678, 679, 680, 681, 682, 683, 684, 685, 686, 687, 688, 689, 690, 691, 692, 693, 694, 695, 696, 697, 698, 699, 700, 701, 702, 703, 704, 705, 706, 707, 708, 709, 710, 711, 712, 713, 714, 715, 716, 717, 718, 719, 720, 721, 722, 723, 724, 725, 726, 727, 728, 729, 730, 731, 732, 733, 734, 735, 736, 737, 738, 739, 740, 741, 742, 743, 744, 745, 746, 747, 748, 749, 750, 751, 752, 753, 754, 755, 756, 757, 758, 759, 760, 761, 762, 763, 764, 765, 766, 767, 768, 769, 770, 771, 772, 773, 774, 775, 776, 777, 778, 779, 780, 781, 782, 783, 784, 785, 786, 787, 788, 789, 790, 791, 792, 793, 794, 795, 796, 797, 798, 799, 800, 801, 802, 803, 804, 805, 806, 807, 808, 809, 810, 811, 812, 813, 814, 815, 816, 817, 818, 819, 820, 821, 822, 823, 824, 825, 826, 827, 828, 829, 830, 831, 832, 833, 834, 835, 836, 837, 838, 839, 840, 841, 842, 843, 844, 845, 846, 847, 848, 849, 850, 851, 852, 853, 854, 855, 856, 857, 858, 859, 860, 861, 862, 863, 864, 865, 866, 867, 868, 869, 870, 871, 872, 873, 874, 875, 876, 877, 878, 879, 880, 881, 882, 883, 884, 885, 886, 887, 888, 889, 890, 891, 892, 893, 894, 895, 896, 897, 898, 899, 900, 901, 902, 903, 904, 905, 906, 907, 908, 909, 910, 911, 912, 913, 914, 915, 916, 917, 918, 919, 920, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000.



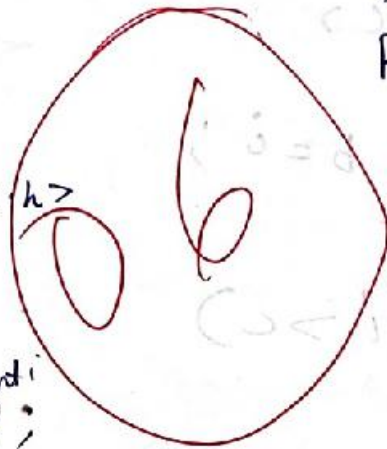
Computer science

VISHNU.T.P

CS-II

Roll-51

```
1) #include <stdio.h>
#include <conio.h>
void main()
{
    int a, int b, int c;
```



```
    printf ("enter the first number");
    scanf ("%d", &a);
```

```
    printf ("enter the second number");
```

```
    scanf ("%d", &b);
```

```
    printf ("enter the third number");
```

```
    scanf ("%d", &c);
```

```
    if (a > b)
```

```
    {
        printf ("a is greater than b");
    }
```

}

```
else (a > c)
```

```
{
    printf ("a is greater of the 3 numbers %d, &a);
```

}

```
return 0;
```

```
printf ("c is greater of the 3 numbers
```



```

    {
        a = i;
    }
else ( )
    {
        b = i;
    }
if ( i > c )
    {
        printf ( "the number %d is greater", i );
    }
else
    {
        printf ( "the number %d is greater", c );
    }
}

```

2)
4)

Tokens

are the smallest part

tokens are of 6 types

- Keyword
- identifier
- Literals
- Operator
- S token
- special characters



Part B

```
#include <stdio.h>
#include <conio.h>
void main ()
```

```
{
    int a, b, c ;
    clrscr();
```

```
    printf ("enter the limit");
```

```
scanf ("%d", &a);
```

```
for (i=0; i<a; i++)
```

Part A

4) Keyword

Key words are the basic building blocks of program. They are predefined words with predefined meanings.

4

eg int, float, char, double etc

Identifier

identifiers are used defined words with user defined meanings.

eg. x, y, z

rules for naming an identifier.

- The first character should be a letter or
- but spaces are not allowed.
- ~~Keywords cannot be used as identifier.~~
- ~~only letters, digits, and special character can be used.~~

Literals

Literals are the constant values contents

Operators

Operators are the ~~are~~ operators which is used for operations.

eg +, -, *, /, %

Strings

They are the character which are enclosed with double quotes "



2CS2

SCMS SCHOOL OF ENGINEERING & TECHNOLOGY

PART - A

01

```

D
#include <stdio.h>
#include <conio.h>
void main ()
{
    int a, b, c;
    clrscr ();
    printf ("enter the numbers")
    scanf ("%d %d %d", &a, &b, &c);
    if (a < b)
        printf ("%d is greater", a);
    else if (b > c)
        printf ("%d is greater", b);
    else
        printf ("%d is greater", c);
    getch ();
    m = i++ && j++ && k++ || --j;
    printf ("%d %d %d %d %d", i, j, k, l, m);
}
    
```

output

2. output

3) Entry Controlled loop

In these, compilation is done before starting the program.

Exit Controlled loop

Compilation is done after running the programme.



4) int

used to represent integer values.

Float

Used to represent floating point numbers.

char

Used to represent characters

Double

It is the greatest of all

long int

PART- B

1) `#include <stdio.h>`
`#include <conio.h>`
`void main()`
`{`
`int x, y, z;`
`* clrscr();`
`printf ("enter the no:");`
~~`scanf ("%d %d %d`~~
~~`&a &b &c`~~
`scanf ("%d %d" &a &b)`
`printf ("%d %d`

2. `#include <stdio.h>`
`#include <conio.h>`
`void main()`
`{`
`int`



SCMS SCHOOL OF ENGINEERING & TECHNOLOGY

Date: 6-03-18

Subject: COMPUTER PROGRAMMING

1- Declaration of 1-D array:

Syntax:

```
datatype arrayname [size];
```

eg: `int a[10];`

1-D array declarations should have datatype of array (datatype of elements contained in array), followed by arrayname and size of the array in square braces, and it should end with a semicolon.

2- Initialization of 1-D array:

(i) During ~~run~~ compile time:

It should have name of the array followed by elements in a curly braces.

eg:

```
int a[10] = {1, 2, 3, 4};
```

```
char a[10] = "Computer";
```

```
char a[10] = {'C', 'O', 'M', 'P', 'U', 'T', 'E', 'R'};
```

(ii) During run time:

scanf can be used to get the input during run time

for eg:

```
printf("Enter the no. of elements");
```

```
scanf("%d", &n);
```

```
for(i=0; i<n; i++)
```

```
{
```

```
scanf("%d", &a[i]);
```

```
}
```

- Declaration of 2-D array:

Syntax:

```
datatype arrayname [rowsize][columnsize];
```

eg: `int a[10][10];`

Row size and column size should be specified while declaring a matrix or 2-D array.

- Initialization of 2-D array:

(i) During compile time:

~~int a[10][10] = { 2, 3, 4, 5, 6 }~~

int a[2][2] = { 2, 2, 4, 6 };

(ii) During runtime:

```
printf("enter the no. of rows");
```

```
scanf("%d", &r);
```

```
printf("enter the no. of columns");
```

```
scanf("%d", &c);
```

```
for(i=0; i<r; i++)
```

```
{
    for(j=0; j<c; j++)
```

```
{
    scanf("%d", &a[i][j]);
}
```

2, Functions are self contained block of codes, which can perform a particular function. Functions reduce the length of program. The debugging of the program is made easy by functions. Operations are made easy just by adding a function call in the program.

Types of functions:

(1) Functions with no parameters ^{or arguments} and no return value ^{arguments}:

These type of functions do not pass any parameter to the called function and do not return any value to the calling function. There is no data transfer between calling and called functions.

for eg:

```
#include <stdio.h>
```

```
#include <conio.h>
```

```
void fact(void);
```

```
void main()
```

```
{
```

```
    fact();
```

```
}
```

```
void fact(void);
```

```
void fact(void)
```

```
{
```

```
    int n, i, f = 1;
```

```
    printf("enter the number");
```

```
    scanf("%d", &n);
```

```
    for(i=1; i<n; i++)
```

```
    {
```

```
        f = f * i;
```

```
    }
```

```
    printf("factorial is %d", f);
```

```
}
```

(2) Functions with arguments and no return value:

These type of function passes arguments to the called function but do not return any value to the calling function

for eg:

```
#include <stdio.h>
```

```
#include <conio.h>
```

```
void fact(int n);
```

```
void main()
```

```
{
```

```
    int n;
```

```
    printf("enter the number");
```

```
    scanf("%d", &n);
```

```
    fact(n);
```

```
void fact(int n)
```

```
{
```

```
    int f = 1;
```

```
    for(i=1; i<n; i++)
```

```
    {
```

```
        f = f * i;
```

```
    }
```


(3) Functions with arguments and return value:

This type of function passes arguments to the calling function and return values to the calling function.

for eg:

```
#include <stdio.h>
#include <conio.h>
int fact(int n);
void main()
{
    int n;
    printf("enter the number");
    scanf("%d", &n);
    y = fact(n);
    printf("factorial is %d", y);
    getch();
}
```

```
int fact(int n)
{
    int i, f = 1;
    for(i = 1; i < n; i++)
        f = f * i;
    return(f);
}
```

(4) Functions with return value but no arguments:

Build functions such as getch() are the examples of these type of functions. They do not pass any arguments but returns a value.

```
eg: getch();
```

(5) Functions with multiple return value:

Normally functions can return only a single value, and if we want to return multiple value, address operator (&) and indirection operator (*) is used.

for eg:

```
#include <stdio.h>
#include <conio.h>
```




SCMS SCHOOL OF ENGINEERING & TECHNOLOGY

```
int mathop(int a, int b, int *s, int *d);
```

```
void main()
```

```
{
  int a, b, s, d;
```

```
  a = 10;
```

```
  b = 20;
```

```
  mathop(a, b, &s, &d);
```

```
  printf("sum is %d", s);
```

```
  printf("diff is %d", d);
```

```
  getch();
```

```
}
```

```
int mathop(int a, int b, int *s, int *d)
```

```
{
```

```
  *s = b + a;
```

```
  *d = b - a;
```

```
}
```

```
3, #include <stdio.h>
```

```
#include <conio.h>
```

```
void main()
```

```
{
  int n, a[10], i, largest, smallest;
```

```
  printf("enter the no. of elements");
```

```
  scanf("%d", &n);
```

```
  printf("enter the elements");
```

```
  for(i=0; i<n; i++)
```

```
  {
    scanf("%d", &a[i]);
```

```
  }
```

```
  largest = a[0];
```

```
  for(i=0; i<n; i++)
```

```
  {
    if(a[i] > largest)
```

```
    largest = a[i];
```

```
  }
```

(5)

```

} printf("largest is %d", largest);

```

```

smallest = a[0];
for(i=0; i<n; i++)
{
  if(a[i] < smallest)
  {
    smallest = a[i];
  }
}

```

```

} printf("smallest is %d", smallest);
getch();
}

```

```

4. #include <stdio.h>
#include <conio.h>

```

```

void main()
{
  int i, j, n, temp;
  char a[5][5];
  printf("enter no. of names");
  scanf("%d", &n);
  printf("enter names");
  for(i=0; i<n; i++)

```

```

  {
for(j=0; j<i; j++)
    scanf("%s", a[i]);

```

```

  }
  for(i=0; i<n; i++)
  {
    for(j=0; j<i; j++)

```

```

      if(strcmp(a[j], a[i]) > 0)

```

```

        strcpy(a[j], temp);
        strcpy(a[j], a[i]);

```

```

int i, temp;
char a[5][5];
for(i=0; i<n; i++)
  printf("no. of names");
  scanf("%d", &n);
  printf("enter names");
  for(i=0; i<n; i++)
  {
    scanf("%s", a[i]);
  }
  for(i=0; i<n; i++)
  {
    for(j=0; j<i; j++)
    {
      if(strcmp(a[j], a[i]) > 0)
      {
        strcpy(a[j], temp);
        strcpy(a[j], a[i]);
        strcpy(a[i], temp);
      }
    }
  }
  for(i=0; i<n; i++)
  {
    printf("%s", a[i]);
  }
  getch();
}

```



```

strcpy(a[j+1], temp);
}
}
for(i=0; i<n; i++)
{
for(j=0; j<i; j++)
{
printf("%0.5s", a[i][j]);
}
}

```

```

getch();
}

```

5, Parameter passing methods :

(1) Call by value :

Here, direct value is passed on to the called functions

for eg:

```

#include <stdio.h>
#include <conio.h>
int fact(int a);
void main()

```

```

{
int n, i;
printf("enter the number");
scanf("%d", &n);
fact(n);
}

```

```

int fact(int a)
{
int i, f=1;
for(i=1; i<a; i++)
f=f*i;
printf("%d", f);
}

```

In this case direct value, n is passed on to the called functions by the caller functions call.

(2) Call by reference
 In this case, address of the value is passed on function instead of the real value.

for eg:

```
#include <stdio.h>
#include <conio.h>
int mathop(int a, int b, int *s, int *d);
void main()
{
  int a, b, s, d;
  a = 10;
  b = 20;
  mathop(a, b, &s, &d);
  printf("sum is %d", s);
  printf("diff is %d", d);
  getch();
}
int mathop(int a, int b, int *s, int *d)
{
  *s = b + a;
  *d = b - a;
}
```

Here address of value is passed by using address operator (&)



- When we are passing address of the value to the function if we make any change to the value, the value of variable itself changes.

```
for(i=0; i<3; i++)
  for(j=0; j<3; j++)
  {
    if(b[a[i]][j] + b[i][j])
      printf("sym");
    else
      printf("not sym");
  }
  b[i][j] = a[i][j];
}
printf("transpose is");
for(i=0; i<3; i++)
  for(j=0; j<3; j++)
    printf("%d", b[i][j]);
```



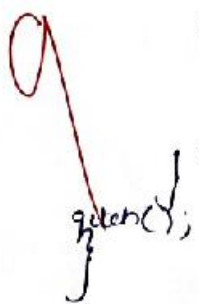
PART-B

```
1, #include <stdio.h>
# include <conio.h>
void main()
{
  int a[5][5], b[5][5], i, j, r, c, d=0;
  printf("enter no of rows");
  scanf("%d", &r);
  printf("enter no of columns");
  scanf("%d", &c);
  for(i=0; i<r; i++)
  {
    for(j=0; j<c; j++)
    {
      scanf("%d", &a[i][j]);
    }
  }
  for(i=0; i<r; i++)
  {
    for(j=0; j<c; j++)
    {
      b[i][j] = a[j][i];
    }
  }
  printf("transpose is\n");
  for(i=0; i<r; i++)
  {
    for(j=0; j<c; j++)
    {
      printf("%d", b[i][j]);
    }
  }
}
```

```

for (i=0; i<n; i++)
{
for (j=0; j<n; j++)
{
if (a[i][j] != b[j][i])
{
printf("matrix is not symmetric");
d = 1;
break;
}
else
d = 0;
}
if (d == 1)
printf("matrix is not symmetric");
else
printf("matrix is not symmetric");
}
}
}

```



```

2, #include <stdio.h>
#include <conio.h>
void main()
{
int n, choice, term, sum, deg, i;
char x;
do
{
printf("1. exponential \t 2. sine series \t 3. cosine series");
printf("enter the choice");
scanf("%d", &choice);
switch (choice)
{
case 1:
printf("enter the number");
scanf("%d", &n);
printf("enter limit");
scanf("%d", &deg);
}
}
}

```

```

void main()
int n, choice, term,
char x;
do
{
printf("1. Exp 2. sine 3. c");
printf("choice");
scanf("%d", &choice);
switch (choice)
{
case 1:
printf("n");
scanf("%d", &n);
term = x;
sum = 1 + x;
for (i=2; i<=n; i++)
{
term = term * x / i;
sum = sum + term;
}
printf("%d", sum);
break;
}
}
}

```

$$1 + x + \frac{x^2}{2!} + \frac{x^3}{3!}$$

$$\frac{x^2}{2} + \frac{x^3}{3} + \frac{x^4}{4} + \dots$$

$$1 + x + \frac{x^2}{2} + \frac{x^3}{3!} + \dots$$



$x = \frac{2}{31} + \frac{2}{51}$ serie -
↓
odd

```

term = x;
sum = 1 + x;
for (i = 1; i < n; i++)
{
    term = term * x / (i + 1);
    sum = sum + term;
}
printf("%d", sum);
break;

```

Case 2:

```

printf("enter the degree");
scanf("%d", &deg);
printf("enter limit");
scanf("%d", &n);
x = deg * (3.14 / 180);
term = x;
sum = x;
for (i = 1; i < n; i++)

```

```

{
    term = (-term) * x * x / ((2 * i) * ((2 * i) + 1));
    sum = sum + term;
}
printf("%d", sum);
break;

```

Case 3:

```

printf("enter the degree");
scanf("%d", &deg);
printf("enter limit");
scanf("%d", &n);
x = deg * (3.14 / 180);
term = x;
sum = x;

```

```

printf("%d", deg);
scanf("%d", &deg);
printf("limit");
scanf("%d", &n);
x = deg * (3.14 / 180);
term = x;
sum = x;
for (i = 1; i < n; i++)
{
    term = term * x * x / (2 * i);
    sum = sum + term;
}
printf("%d", sum);

```

< d o i b l s > shubam # 18
< d o i n a s > shubam #
() n i s u m k o s

printf("%d", sum);
(2 * i) * ((2 * i) + 1);
(2 * i);

printf("%d", deg);
(n);
(++i; i > 1; 0);
(++i; i > 1; 0);
(d * x / (2 * i));
(++i;

```

for (i=1; i<n; i++)
{
    term = (-term) * 2 * i / ((2 * i) * ((2 * i) - 1));
    sum = sum + term;
}
printf("u%od", sum);
break;
printf("Do you want to continue if yes press 'Y' else 'N'");
scanf(" %c", &z);
while (z == 'Y');
getch();
}

```

```

3, #include <stdio.h>
#include <conio.h>
void main()
{
    int i, k=1, c, j, s, n, choice;
    char z;
    do
    {
        printf("1. Floyd's \t 2. Pascal's");
        scanf("%d", &choice);
        printf("enter the choice");
        scanf("%d", &choice);
        switch (choice)
        {
            case 1:
                printf("enter no. of rows");
                scanf("%d", &n);
                for (i=0; i<n; i++)
                {
                    for (j=0; j<i; j++)
                    {
                        printf("%d\t", k);
                        k++;
                    }
                }
            }
        }
    } while (choice != 0);
}

```

```

Floyd's Δ
int i, k=1, j, s;
printf("no. of rows");
scanf("%d", &n);
for (i=0; i<n; i++)
{
    for (j=0; j<i; j++)
    {
        printf("%d\t", k);
        k++;
    }
    printf("\n");
}
Pascal's
printf("no. of rows");
scanf("%d", &n);
for (i=0; i<n; i++)
{
    for (j=0; j<=n-i; j++)
    {
        printf(" ");
    }
    for (j=0; j<i; j++)
    {
        if (i==0 || j==0)
            c=1;
        else
            c=c*(i-j)/j;
        printf("%d\t", c);
    }
}

```



SCMS SCHOOL OF ENGINEERING & TECHNOLOGY

My

```
printf("\n");
```

```
}
```

```
break;
```

case 2:

```
printf("enter no. of rows");
```

```
scanf("%d", &n);
```

```
for(i=0; i<n; i++)
```

```
{
```

```
for(s=1; s<=n-i; s++)
```

```
printf(" ");
```

```
}
```

```
for(j=0; j<i; j++)
```

```
{ if(i==0 || j==0)
```

```
c=1;
```

```
else
```

```
c=c*(i-j+1)/j;
```

```
printf("%d", c);
```

```
}
```

```
printf("\n");
```

```
}
```

```
break;
```

```
printf("Do you want to continue, if yes press 'y' else 'n');
```

```
scanf("%c", &z);
```

```
while(z != 'y');
```

```
getch();
```

```
}
```

10

PART A

1, Algorithm:

Step 1: Start

Step 2: Get the input rows and columns

Step 3: Get the input matrix

Step 4: Repeat the steps until $i < r$ is reached

4.1: Repeat the steps until $j < c$ is reached

4.2: $b[i][j] \leftarrow a[j][i]$

Step 5: Display $b[i][j]$

Step 6: Repeat steps until $i < r$ is reached

6.1: Repeat steps until $j < c$ is reached

6.2: Check whether $a[i][j] == b[i][j]$ if true go to 6.3 else go to 6.4

6.3: $d \leftarrow 1$;

6.4: $d \leftarrow 0$;

7: Check whether $d == 0, 1$; if true go to 7.1 else to 7.2

7.1: Display matrix is symmetric

7.2: Display matrix is not symmetric

8: Stop

PART-A

3, step 1: Start

2: Get the input ~~rows~~ and elements

3: $largest \leftarrow a[0]$

4: Repeat the steps until $i < n$ is reached

4.1: Check whether $a[i] > largest$, if true go to 4.2

4.2: $largest \leftarrow a[i]$

4.5: Display largest

6: $smallest \leftarrow a[0]$

7: Check whether $a[i] < smallest$, if true go to 7.1

7.1: $smallest \leftarrow a[i]$

8: Display smallest

9: stop



SCMS SCHOOL OF ENGINEERING & TECHNOLOGY

Maniya Raphael
 Branch & Roll No. CS2, 3
 Date: 6-3-2019

Subject: Computer Programming

PART B

Menu driven program for floyds and pascals triangle



```
#include <stdio.h>
#include <conio.h>
void main()
{
  char ch;
  int n, k, i, j;
  do
  {
    printf("1. Floyds Triangle \n 2. Pascals Triangle");
    printf("\n Enter your choice");
    scanf("%d", &ch);
    switch(ch)
    {
      // FLOYDS
      case 1 :
        printf("Enter the number of rows\n");
        scanf("%d", &n);
        printf("Floyds Triangle");
        k = 1;
        for(i=1; i<=n; i++)
        {
          printf("\n");
          for(j=1; j<=i; j++)
          {
            printf("\t%d", k);
            k = k+1;
          }
        }
        break;
    }
  }
}
```

i=2.
→ 1
2 3.

Case 2:
printf("Enter number of rows");

scanf("%d", &n);

printf("in Pascal Triangle");

for(i=0; i<n; i++)

{

for(k=1; k<=n-i; k++)

{ printf(" ");

for(j=0; j<=i; j++)

{

if(i==0 || j==0)

c=1;

else

c=c*(i-j+1)/j;

printf("%d", c);

}

printf("\n");

break; }

default:

printf("Sorry! Invalid choice");

break;

}

printf("Do you want to continue? Press y or n");

scanf("%c", &d);

} while (d == 'y');

getch();

}

10

Output

- 1. Floyd's Triangle
- 2. Pascal's Triangle

Enter your choice 1

Enter number of rows 4

Floyd's Triangle

```
1
2 3
4 5 6
7 8 9 10
```

Do you want to continue? Press y or n
y

- 1. Floyd's Triangle
- 2. Pascal's Triangle

Enter your choice 2

Enter number of rows 3

Pascal Triangle

```
1
1 1
1 2 1
```

// Transpose and Symmetry

```
1) #include <stdio.h>
#include <conio.h>
void main()
{
int a[10][10], b[10][10], i, j, n, c, k=0;
clrscr();
printf("Enter no of rows");
scanf("%d", &n);
printf("Enter no of columns");
scanf("%d", &c);
if(n==c)
printf("\n Enter the elements of matrix");
for(i=0; i<n; i++)
{
for(j=0; j<c; j++)
{
scanf("%d", &a[i][j]);
}
}
}
```

// Transpose

```
for(i=0; i<n; i++)
{
for(j=0; j<c; j++)
{
b[i][j] = a[j][i];
}
}
```





SCMS SCHOOL OF ENGINEERING & TECHNOLOGY

```
// symmetry check
for(i=0; i<n; i++)
{
  for(j=0; j<c; j++)
  {
```

```
a[0][0] = 1
a[0][1] = 2
a[0][2] = 5
a[1][0] = 2
a[1][1] = 6
a[1][2] = 7
a[2][0] = 5
a[2][1] = 7
a[2][2] = 9
```

```
if (a[i][j] != b[j][i])
```

```
1 0 1
1 2 5
2 6 7
5 7 9
```

```
    k = 1;
    break;
```

```
if (k == 1)
```

```
    printf("The given matrix is not symmetric");
```

```
else
```

```
    printf("The given matrix is symmetric");
```

```
} }
```

```
getch();
```

```
else
```

```
    printf("The matrix is not square and not symmetric");
```

Output

```
Enter no. of rows }
getch(); }
```

```
1 0 1
2 6 7
5 7 9
```


Output

I Enter no. of rows 3
Enter no. of columns 3
Enter the elements of matrix
1 2 5 2 6 7 5 7 9
The given matrix is symmetric.

II Enter no. of rows 2
Enter no. of columns 3
The matrix is not square and not symmetric.

2) Menu driven - Sine Series, Cosine Series, Exponential Series

```
#include <stdio.h>
```

```
#include <conio.h>
```

```
#include <math.h>
```

```
void main()
```

```
{ long int fact;
```

```
int ch, x, series n, num, i, j, temp, t;
```

```
float term, sum, y;
```

```
char d;
```

```
do
```

```
{
```

```
printf("\n 1. Exponential Series\n 2. Sine Series\n 3. Cosine Series");
```



SCMS SCHOOL OF ENGINEERING & TECHNOLOGY

Name: Umesh C. Arid
Branch & Roll No. CS II, 4, 52
Date: 06.08.2018

Subject: Computer Programming

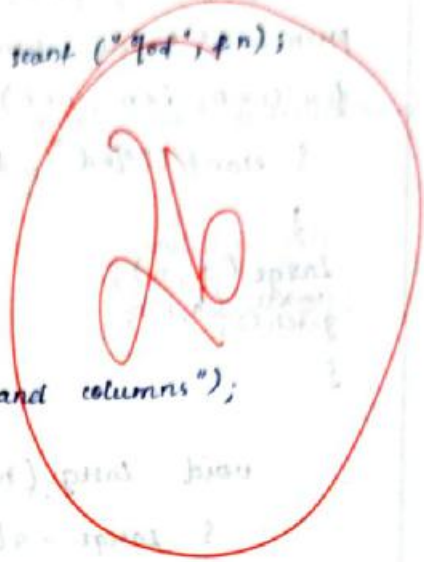
PART A

1. 1-d array.

```
int a[10];
printf("Enter no. of elements"); scanf("%d", &n);
for (i=0; i<n; i++)
    { scanf("%d", &a[i]);
    }
```

2-d array

```
int a[10][20];
printf("Enter no. of rows and columns");
scanf("%d %d", &x, &y);
for (i=0; i<x; i++)
    { for (j=0; j<y; j++)
        { scanf("%d", &a[i][j]);
        }
    }
```



2. Function is a module or a subprogram with a set of statements for executing a particular task. It can be called by the main program infinite no. of times.

```
void greet (int x)
{ int n = 2;
  if (x == n)
    { printf("Hello");
    }
  else
    { printf("Bye");
    }
}
```

Function is divided into two:

- predefined function
- userdefined function

Predefined functions are defined in the library.

eg: printf(), scanf() etc.

Userdefined functions are defined by the programmer (user)

```
eg: void multiply ()
{ a=5; b=6;
  printf("%d", a*b);
}
```

```

#include <stdio.h>
#include <conio.h>
void lastge (int a[], int n);
void main ()
{
    int a[50], n;
    clrscr();
    printf ("Enter the no. of elements");
    scanf ("%d", &n);
    printf ("Enter elements into array");
    for (i=0; i<n; i++)
        { scanf ("%d", &a[i]);
        }
    large (a, n);
    small (a, n);
    getch();
}

```

```

void lastge (int a[], int n)
{
    large = a[0];
    for (i=1; i<n; i++)
        { if (a[i] > large)
            { large = a[i];
            }
        }
    printf ("%d is largest", large);
}

```

```

void small (int a[], int n)
{
    small = a[0];
    for (i=1; i<n; i++)
        { if (a[i] < small)
            { small = a[i];
            }
        }
    printf ("%d is smallest", small);
}

```


2)

(1) Continuation
Function with no arguments and no return value.

Function definition:
void multiply (void)

```
{ int a = 5;
  int b = 10;
  y = a * b;
  printf("%d", y);
}
```

Function call
multiply ();

(2) Function with arguments and no return value.

Function definition:

```
void add (int x)
{ int y = 10;
  int p;
  p = y + x;
  printf("%d", p);
}
```

Function call
add (p);

(3) Function with arguments and return value.

Function definition:

```
int multiply (int x)
{ int a = 5;
  int y;
  y = a * x;
  return (y);
}
```

Function call
multiply (p);

~~(4) Function with no~~

5) The two different parameter passing methods are:

- Passing by value.
- Passing by reference.

Passing by value:

In this case, a copy of the actual parameters is sent to the formal parameters. Changes done to the formal parameters will not affect the actual parameters.

```

void add (int x, int y);
void main ()
{
  int a, b;
  printf ("Enter two numbers");
  scanf ("%d %d", &a, &b);
  add (a, b);
  getch();
}

```

```

void add (int x, int y)
{
  int p;
  p = x + y;
  x = x + 1;
  y = y + 1;
  printf ("%d", p);
}

```

Here, x & y gets incremented by 1 but a & b remain same.

passing by reference:

Here the address of the actual parameters is send to the called function. Changes done to the variables in the called function will affect the actual parameters.

same main function

```

void add (int *x, int *y)
{
  int p;
  p = *x + *y;
  *x = *x + 1;
  *y = *y + 1;
}

```

function call

```
add (&a, &b);
```

function declaration

```
void add (int *a, int *b);
```

Here, on changing the values of x & y, a & b gets affected.



PART B

```

1. #include <stdio.h>
#include <conio.h>

void main()
{ int a[10][10], b[10][10], m, n, j, i, c=0;
  clrscr();

```

```

  printf("Enter the no. of rows and columns");
  scanf("%d %d", &m, &n);
  printf("Enter elements");
  for(i=0; i<m; i++)

```

```

    { for(j=0; j<n; j++)
      { scanf("%d", &a[i][j]);

```

```

        }
      }
    }
  }
  for(i=0; i<m; i++)
  { for(j=0; j<n; j++)
    { b[i][j] = a[j][i];

```

```

    }
  }
  printf("transpose of matrix");
  for(i=0; i<m; i++)
  { for(j=0; j<n; j++)
    { printf("%d", b[i][j]);

```

```

  }
  }
  if (m == n)
  { for(i=0; i<m; i++)
    { for(j=0; j<n; j++)
      { if(a[i][j] != b[i][j])
        { c=1;
          break;
        }
      }
    }
  }

```

```

  if (c == 0)
  { printf("It is symmetric");
  }
  else
  { printf("It is not symmetric");
  }

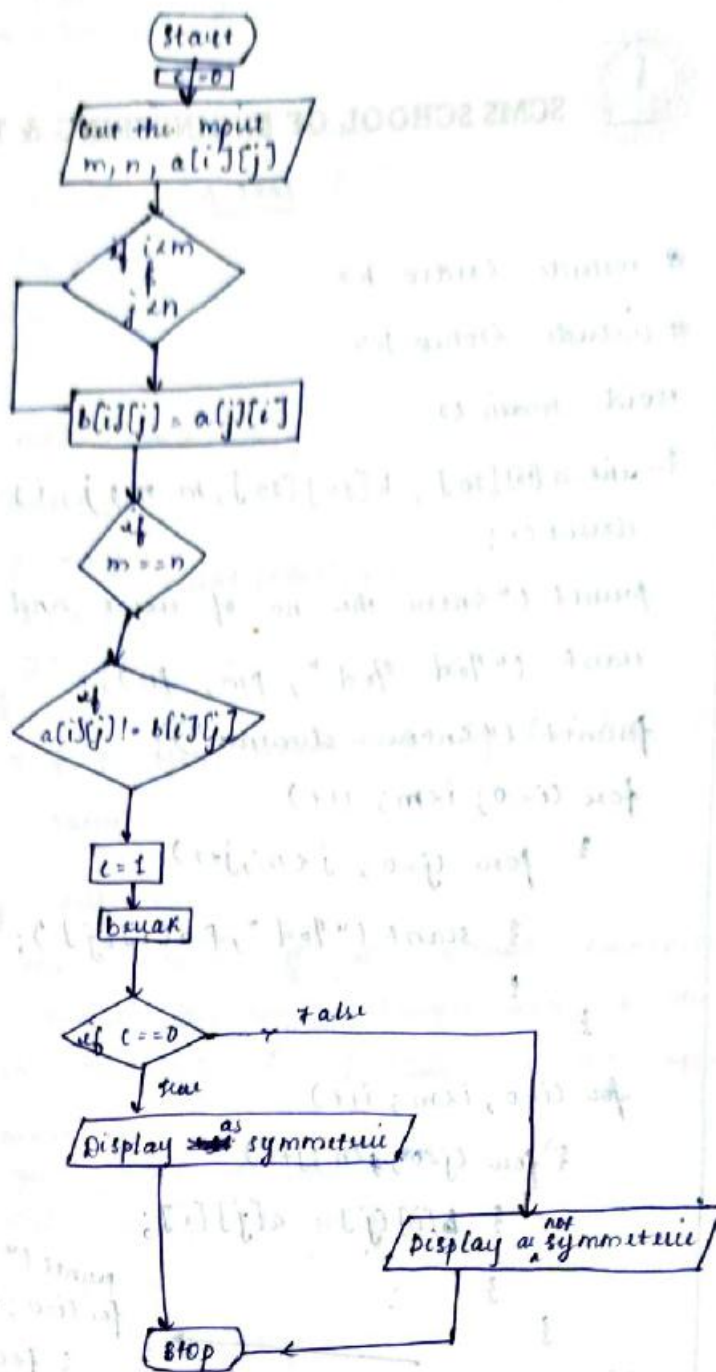
```

```

  getch();
}

```

GY



```

3. (i) #include <stdio.h>
        #include <conio.h>
        void main()
        { int n, i, j;
          clrscr();
          printf("enter no. of rows");
          scanf("%d", &n);
          for (i=1; i<=n; i++)
            { printf("%d", i);
  
```

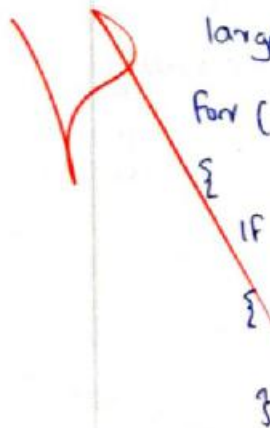
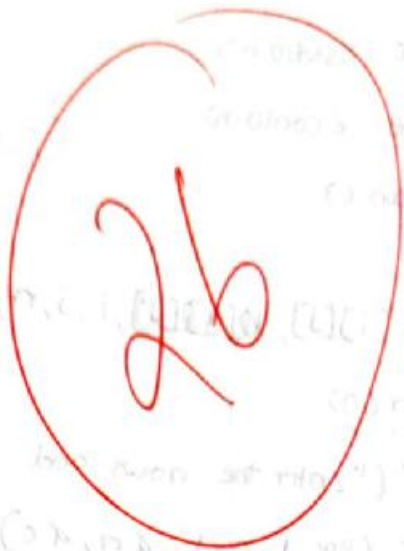
PART A

3.

```

#include <stdio.h>
#include <conio.h>
void main()
{
int a[n], i, j, n;
clrscr();
printf("Enter the limit");
scanf("%d", &n);
printf("Enter the elements");
for(i=0; i<n; i++)
{
scanf("%d", &a[i]);
}
smallest = a[0];
// smallest -> a[i]
for(i=0; i<n; i++)
{
if(smallest > a[i])
{
smallest = a[i];
}
}
printf("smallest is", smallest);
}
largest = a[0];
for(i=0; i<n; i++)
{
if(largest < a[i])
{
largest = a[i];
}
}
printf("largest is", largest);
}

```



```
getch(c);  
}
```

PART B
1.

Transpose of a matrix:

```
#include <stdio.h>  
#include <conio.h>  
void main (c)  
{  
    int a[i][j], a[i][j], i, j, r, c;  
    clrscr(c);  
    printf ("Enter the rows and columns");  
    scanf ("%d %d", &r, &c);  
    printf ("Enter the elements");  
    for(i=0; i<r; i++)  
    {  
        for(j=0; j<c; j++)  
        {  
            scanf ("%d", &a[i][j]);  
        }  
    }  
    for(i=0; i<r; i++)  
    {  
        for(s=0; s<c; s++)  
        {  
            printf (" a [%d][%d] ", i, s);  
        }  
    }  
    getch(c);  
}
```

given matrix is symmetric or not?

```
#include <stdio.h>  
#include <conio.h>  
void main (c)  
{  
    int a[i][j], b[i][j], i, j, r, c;
```



```
check();
```

```
printf("Enter the rows and columns");
```

```
scanf("%d %d", &r, &c);
```

```
for (i=0; i<r; i++)
```

```
{
```

```
for (j=0; j<c; j++)
```

```
{
```

```
scanf("%d", &a[i][j]);
```

```
}
```

```
}
```

```
for (i=0; i<r; i++)
```

```
{
```

```
for (j=0; j<c; j++)
```

```
{
```

```
b[i][j] = a[j][i];
```

```
}
```

```
}
```

```
if (b[i][j] != a[i][j])
```

```
{
```

```
flag = 1;
```

```
break;
```

```
}
```

```
if (flag == 1)
```

```
{
```

```
printf("The given matrix is not symmetric");
```

```
}
```

```
else
```

```
{
```

```
printf("The given matrix is not symmetric");
```

```
}
```

```
getch();
```

```
}
```

expat
(/n);

PART B

2. menu driven program to find the sum of
- i) sine series
 - ii) cosine series
 - iii) exponential series

```
#include <stdio.h>
#include <conio.h>

void main()
{
    int i, n, condition, fact;
    float term, sum, deg;
    char ch;

    do
    {
        printf("1. Exponential | 2. Sine series | 3. Cosine series |");
        scanf("%d", &condition);
        printf("Enter the choice");
        scanf("%d", &condition);

        switch (condition)
        {
            case 1:
                printf("Enter the value for n");
                scanf("%d", &n);
                printf("Enter the value for r");
                scanf("%d", &n);

                term = 1;
                fact = 1;
                sum = 1;
                sum = 1;

                for (i = 1; i <= n; i++)
                {
                    fact = fact * i;
                    sum =
                    term = (term * r) / fact;
                    sum = sum + term;
                }
                break;
        }
    }
}
```



Case 2:

```

printf ("Enter the value for x");
scanf ("%d", &deg);
printf ("Enter the value for n");
scanf ("%d", &n);
x = deg * 0.01744;
term = x;
sum = x;
for (i=0; i<n; i++)
{
  term = (-1)i * term * x * x / ((2+i) * (2+i));
  sum = sum + term;
}
break;

```

Case 3:

```

printf ("Enter the value for n");
scanf ("%d", &deg);
printf ("Enter the value for n");
scanf ("%d", &n);
x = deg * 0.01744;
term = 1;
sum = 1;
for (i=0; i<n; i++)
{
  term = term * x * x / ((2+i) * (2+i-1));
  sum = sum + term;
}
break;

```

10

PA

4. Program to solve a set of names

#include <stdio.h>

#include <conio.h>

void main()

{

while

{

do print ("do you want to continue Y or N")

if scanf ("%d", &c);

if (c == 'Y');

{

}

else {

break;

}

getch();

}

PART A

1. declaration and initialisation of 1-d and 2-d array

declaration of one-dimensional array

for ex: a[i][j]

⇒ a[i][j]

where a = array name

[i] - number of elements in the array

declaration of two dimension array

⇒ a[i][j]

where a = array name

i - number of rows

j - number of column.

int a[i][j]

initialisation of one-dimension array :-

complete time we can use



PART-B
3.

Menu driven program to display floyds and pascals triangle

```
#include <stdio.h>
```

```
#include <conio.h>
```

```
void main()
```

```
{
```

```
int i, j, condition;
```

```
do
```

```
{
```

```
printf (" 1. floyds triangle 2. pascals triangle ");
```

```
printf ("Enter the choice ");
```

```
scanf ("%d", &condition);
```

```
switch (condition)
```

```
{
```

```
case 1:
```

floyds

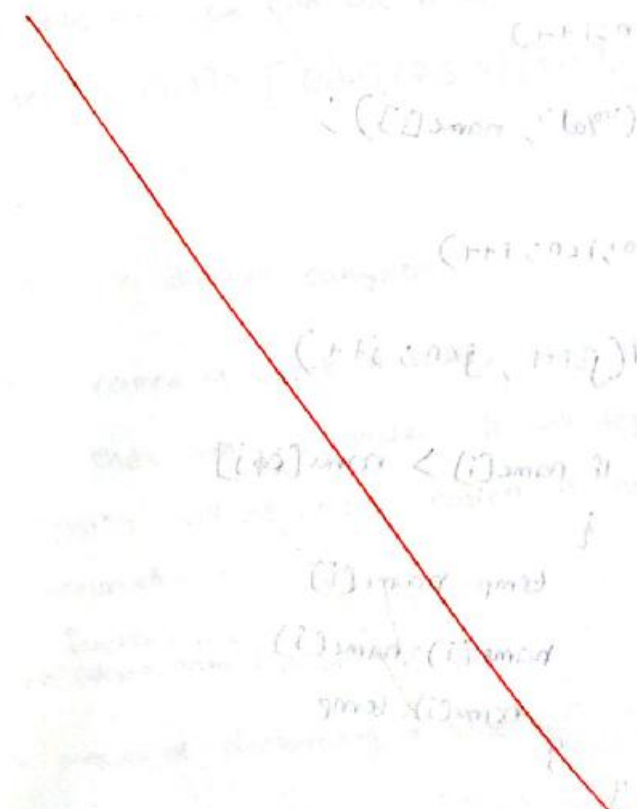
```
1
2 3
4 5 6
```

pascals

```
1
1 2 1
1 2 3 1
```

```
1 2
2 3 3
4 5 6 6
7 8 9
```

n=1 x 2
n= 3 x 4
= 4 5 6 6
n= 2 x 4
= 7





SCMS SCHOOL OF ENGINEERING & TECHNOLOGY

Subject computers

VISHNUT.P

CSE

Roll - 51

Part B

```

2) #include <stdio.h>
   #include <conio.h>
   void main()

```

```

{
  int x, n, i;

```

```

  Print @ degrees
  (" 1. sine series / n 2. cosine series / n 3. exponential / n);

```

```

  scanf ("%d", &ch);

```

```

  switch (ch)

```

```

  { case 1:

```

```

    float x, temp = 0, sum = 0;

```

```

    printf ("enter the value of x and n");

```

```

    scanf ("%d %d", &x &n);

```

```

    x1 = x * (3.14 / 180);

```

```

    temp = x1;

```

```

    sum = x1;

```

```
for (i=1; i < m; i++)
```

```
{
```

```
temp = temp * x * x * -1 / (a_i * (2i+1));
```

```
sum = sum + temp;
```

```
}
```

```
printf ("%d", sum);
```

```
}
```

```
break;
```

case 2:

```
float x, temp = 0, sum = 0;
```

```
printf ("enter the value of x and m");
```

```
scanf ("%d %d", &x, &m);
```

```
x = x * (3.14 / 180);
```

```
temp = 1;
```

```
sum = 1;
```

```
for (i=1; i < m; i++)
```

```
{
```

```
temp = temp * x * -1 / (a_i * (2i-1));
```

sum = sum + temp;

}

printf (" %d", sum);

}

break;

case 3 :

temp = 0, sum = 0;
float x, m;

printf (" enter the value of x and m");

scanf (" %d %d", &x, &m);

~~x = x * (3.14 / 180);~~

temp = x;

sum = 1 + x;

f = 1;

for (i = 1; i < m; i++)

{

temp = temp * x / f * i;

sum = sum + temp;

}


```
Print f ("%d", sum);
```

```
}
```

```
break;
```

```
Print f ("do you want to continue " );
```

```
scanf ("%d", &ch);
```

```
}
```

algorithm

Step 1: start

Step 2: initialise variables.

Step 3: get desired count ~~from~~ ^{from user.}

Step 4: if output is 1
go to step 5

Step 5: $temp = temp * x * x * -1 (2i (2i + 1))$
 $sum = sum + temp$

Step 6: if output is 2
go to step 7

Step 7: $temp = temp * x * -1 (2i (2i - 1))$
 $sum = sum + temp$

Step 8: if output is 3
go to step 9

Step 9: $temp = temp * x / f * i$
 $sum = temp + temp$

Step 10: print sum

Step 11: stop

X-----X



SCMS SCHOOL OF ENGINEERING & TECHNOLOGY

Subject COMPUTER PROGRAMMING



1. 1-D array

Declaration

~~data arrayname [size];~~

datatype array name [size];

eg:- int a [50];

3. #include <stdio.h>

#include <conio.h>

void main ()

{
int i, n, a [50], largest, smallest;
clrscr ();

printf ("Enter the no of elements in the array");
scanf ("%d", &n);

for (i=0; i < n; i++)

{
scanf ("%d", &a [i]);

Print f ("%d", i, a[i])

SCHOOL OF
ENGINEERING & TECHNOLOGY



4.

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int n, i, a[50], j, temp;
    clrscr();
    printf("Enter the no of names");
    scanf("%d", &n);
    printf("Enter the names");
    for(i=0; i<n; i++)
    {
        scanf("%d", &a[i]);
    }
    for(i=0; i<n; i++)
    {
        for(j=0; j<n; j++)
```



```

{
    if(a[j] < a[j+1])
    {
        temp = a[j];
        a[j] = a[j+1];
        a[j+1] = temp;
    }
    printf("names after swapping is:");
    {
for(i=0; i<n; i++)
        for(i=0; i<n; i++)
            printf("%d", a[i]);
    }
    getch();
}

```

PART - B

```

2. #include <stdio.h>
#include <conio.h>
void main()
{
    int i, x, term = 0, sum = 0, n, choice;
    float x;
    printf("1. Sine series \n 2. Cosine Series \n 3. Exponential series");
scanf("%d", &choice);
    printf("Enter the choice");
    scanf("%d", &choice);
    do:
    {
        switch(choice)
        {

```

Case 1:

```
printf("Enter the limit");
scanf("%d", &n);
printf("Enter the value of x");
scanf("%d", &x);
x1 = x * (3.14/180);
sum = x1;
term = x1;
for(i=0; i<=n; i++)
{
    term = -term * x1 * x1 / 2i(2i+1);
    sum = sum + term;
}
printf("sum of sine series is %d", sum);
```

Case 2: printf("Enter the limit");

scanf("%d", &n);

printf("Enter the value of x");

scanf("%d", &x);

x₁ = x * (3.14/180);

sum = 1;

term = 1;

for(i=0; i<=n; i++)

{
term = -term * x₁ * x₁ / 2i(2i-1)

sum = sum + term;

}

printf("sum of cosine series is %d", sum);
}

Case 3: