

## INDEX

Sl.No	Contents	Page No
1	Syllabus	i
2	CO – PO Mapping	ii
3	Lab assessment rubrics for CIE	iii
4	Introduction to Principals of Programming Using C	1
5	Simulation of a Simple Calculator	4
6	Compute the roots of a quadratic equation by accepting the coefficients. Print appropriate messages.	6
7	An electricity board charges the following rates for the use of electricity: for the first 200 units 80 paise per unit: for the next 100 units 90 paise per unit: beyond 300 units Rs 1 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 name of the user, number of units consumed and print out the charges.	8
8	Write a C Program to display the following by reading the number of rows as input, <pre> 1 1 2 1 1 2 3 2 1 1 2 3 4 3 2 1 ----- nth row</pre>	10
9	Implement Binary Search on Integers.	12
10	Implement Matrix multiplication and validate the rules of multiplication	14
11	Compute sin(x)/cos(x) using Taylor series approximation. Compare your result with the built-in library function. Print both the results with appropriate inferences	16
12	Sort the given set of N numbers using Bubble sort	18
13	Write functions to implement string operations such as compare, concatenate, and find string length. Use the parameter passing techniques	20
14	Implement structures to read, write and compute average- marks of the students, list the students scoring above and below the average marks for a class of N students	23
15	Develop a program using pointers to compute the sum, mean and standard deviation of all elements stored in an array of N real numbers	26
16	Write a C program to copy a text file to another, read both the input file name and target file name.	28
17	Viva Voice Questions	30

## SYLLABUS

<b>Course Title:</b>	<b>Principles of Programming using C</b>	
<b>Course Code:</b>	<b>22POP13</b>	<b>CIE Marks 50</b>
<b>Course Type (Theory/Practical /Integrated)</b>	<b>Integrated</b>	<b>SEE Marks 50</b>
		<b>Total Marks 100</b>
<b>Teaching Hours/Week (L:T:P: S)</b>	<b>2:0:2</b>	<b>Exam Hours 3+2</b>
<b>Total Hours of Pedagogy</b>	<b>40 hours</b>	<b>Credits 03</b>
	<b>Course Objectives:</b> CLO 1. Elucidate the basic architecture and functionalities of a computer CLO 2. Apply programming constructs of C language to solve the real-world problems CLO 3. Explore user-defined data structures like arrays, structures and pointers in implementing solutions to problems CLO 4. Design and Develop Solutions to problems using structured programming constructs such as functions and procedures	
	<b>Teaching-Learning Process (General Instructions)</b> These are sample Strategies, which teachers can use to accelerate the attainment of the various course outcomes. <ol style="list-style-type: none"> <li>1. Lecturer method (L) need not to be only traditional lecture method, but alternative effective teaching methods could be adopted to attain the outcomes.</li> <li>2. Use of Video/Animation to explain functioning of various concepts.</li> <li>3. Encourage collaborative (Group Learning) Learning in the class.</li> <li>4. Ask at least three HOT (Higher order Thinking) questions in the class, which promotes critical thinking.</li> <li>5. Adopt Problem Based Learning (PBL), which fosters students' Analytical skills, develop design thinking skills such as the ability to design, evaluate, generalize, and analyze information rather than simply recall it.</li> <li>6. Introduce Topics in manifold representations.</li> <li>7. Show the different ways to solve the same problem and encourage the students to come up with their own creative ways to solve them.</li> <li>8. Discuss how every concept can be applied to the real world-and when that's possible, it helps to improve the students' understanding.</li> <li>9. Use <a href="https://pythontutor.com/visualize.html#mode=edit">https://pythontutor.com/visualize.html#mode=edit</a> in order to visualize the operations of C Programs</li> </ol>	
	<b>Module-1 (6 Hours of Pedagogy)</b>	
	<b>Introduction to C:</b> Introduction to computers, input and output devices, designing efficient programs. Introduction to C, Structure of C program, Files used in a C program, Compilers. Compiling and executing C programs, variables, constants, Input/output statements in C,  <b>Textbook: Chapter 1.1-1.9, 2.1-2.2, 8.1 - 8.6 ,9.1-9.14</b>	
<b>Teaching-Learning Process</b>	Chalk and talk method/Power Point Presentation/ Web Content: <a href="https://tinyurl.com/4xmrexre">https://tinyurl.com/4xmrexre</a>	

	<b>Module-2 (6 Hours of Pedagogy)</b>
Operators in C, Type conversion and typecasting. <b>Decision control and Looping statements:</b> Introduction to decision control, Conditional branching statements, iterative statements, nested loops, break and continue statements, goto statement. <b>Textbook: Chapter 9.15-9.16, 10.1-10.6</b>	
<b>Teaching-Learning Process</b>	Chalk and talk method/ Power Point Presentation
	<b>Module-3 (8 Hours of Pedagogy)</b>
Functions: Introduction using functions, Function definition, function declaration, function call, return statement, passing parameters to functions, scope of variables, storage classes, recursive functions. Arrays: Declaration of arrays, accessing the elements of an array, storing values in arrays, Operations on arrays, Passing arrays to functions, two dimensional arrays, operations on two-dimensional arrays, two-dimensional arrays to functions, multidimensional arrays, applications of arrays. <b>Textbook: Chapter 11.1-11.10, 12.1-12.10, 12.12</b>	
<b>Teaching-Learning Process</b>	Chalk and talk method/ Power Point Presentation
	<b>Module-4 (6 Hours of Pedagogy)</b>
<b>Strings and Pointers:</b> Introduction, string taxonomy, operations on strings, Miscellaneous string and character functions, arrays of strings. <b>Pointers:</b> Introduction to pointers, declaring pointer variables, Types of pointers, Passing arguments to functions using pointers <b>Textbook: Chapter 13.1-13.6, 14-14.7</b>	
<b>Teaching-Learning Process</b>	Chalk and talk method/Power Point Presentation
	<b>Module-5 (6 Hours of Pedagogy)</b>
<b>Structure, Union, and Enumerated Data Type:</b> Introduction, structures and functions, Unions, unions inside structures, Enumerated data type. <b>Files:</b> Introduction to files, using files in C, reading and writing data files. , Detecting end of file <b>Textbook: Chapter 15.1 – 15.10, 16.1-16.5</b>	
<b>Teaching-Learning Process</b>	Chalk and talk method/Power Point Presentation
<b>Course Outcomes(Course Skill Set)</b> At the end of the course the student will be able to:  CO1. Elucidate the basic architecture and functionalities of a computer and also recognize the hardware parts.  CO 2. Apply programming constructs of C language to solve the real world problem  CO 3. Explore user-defined data structures like arrays in implementing solutions to problems like searching and sorting  CO 4. Explore user-defined data structures like structures, unions and pointers in implementing solutions  CO5. Design and Develop Solutions to problems using modular programming constructs using functions	

## Programming Assignments

1 Simulation of a Simple Calculator.

2 Compute the roots of a quadratic equation by accepting the coefficients. Print appropriate messages.

3 An electricity board charges the following rates for the use of electricity: for the first 200 units 80 paise per unit :for the next 100 units 90 paise per unit: beyond 300 units Rs 1 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 name of the user, number of units consumed and print out the charges.

4. Write a C Program to display the following by reading the number of rows as input,

```
      1
    1 2 1
  1 2 3 2
    1
  1 2 3 4 3 2 1
-----
nth row
```

5 Implement Binary Search on Integers.

6 Implement Matrix multiplication and validate the rules of multiplication.

7 Compute  $\sin(x)/\cos(x)$  using Taylor series approximation. Compare your result with the built-in library function. Print both the results with appropriate inferences.

8 Sort the given set of N numbers using Bubble sort.

9 Write functions to implement string operations such as compare, concatenate, and find string length. Use the parameter passing techniques.

10 Implement structures to read, write and compute average- marks of the students, list the students scoring above and below the average marks for a class of N students.

11 Develop a program using pointers to compute the sum, mean and standard deviation of all elements stored in an array of N real numbers.

12. Write a C program to copy a text file to another, read both the input file name and target filename.

Note:

SEE marks for the practical course is 50 Marks.

SEE shall be conducted jointly by the two examiners of the same institute, examiners are appointed by the University

All laboratory experiments are to be included for practical examination.

(Rubrics) Breakup of marks and the instructions printed on the cover page of the answer script to be strictly adhered to by the examiners. OR based on the course requirement evaluation rubrics shall be decided jointly by examiners.

Students can pick one question (experiment) from the questions lot prepared by the internal /external examiners jointly.

Evaluation of test write-up/ conduction procedure and result/viva will be conducted jointly by examiners.

General rubrics suggested for SEE are mentioned here, writeup-20%, Conduction procedure and result in -60%, Viva-voce 20% of maximum marks. SEE for practical shall be evaluated for 100 marks and scored marks shall be scaled down to 50 marks (however, based on course type, rubrics shall be decided by the examiners)

Students can pick one experiment from the questions lot with equal choice to all the students in a batch. Student should develop an algorithm, program, execute and demonstrate the results with appropriate output for the given problem.

Change of experiment is allowed only once and 15% Marks allotted to the procedure part to be made zero.

### **The duration of SEE is 02 hours**

#### **Assessment Details (both CIE and SEE)**

The weightage of Continuous Internal Evaluation (CIE) is 50% and for Semester End Exam (SEE) is 50%. The minimum passing mark for the CIE is 40% of the maximum marks (20 marks out of 50). The minimum passing mark for the SEE is 35% of the maximum marks (18 marks out of 50). A student shall be deemed to have satisfied the academic requirements and earned the credits allotted to each subject/ course if the student secures not less than 35% (18 Marks out of 50) in the semester-end examination(SEE), and a minimum of 40% (40marks out of 100) in the sum total of the CIE (Continuous Internal Evaluation) and SEE (Semester End Examination) taken together.

#### **Continuous Internal Evaluation(CIE):**

##### **Two Unit Tests each of 20 Marks (duration 01 hour)**

- First test after the completion of 30-40 % of the syllabus
- Second test after completion of 80-90% of the syllabus

One Improvement test before the closing of the academic term may be conducted if necessary. However best two tests out of three shall be taken into consideration.

##### **Two assignments each of 10 Marks**

The teacher has to plan the assignments and get them completed by the students well before the closing of the term so that marks entry in the examination portal shall be done in time. Formative (Successive) Assessments include Assignments/Quizzes/Seminars/ Course projects/Field surveys/ Case studies/ Hands-on practice (experiments)/Group Discussions/ others. . The Teachers shall choose the types of assignments depending on the requirement of the course and plan to attain the Cos and POs. (to have a less stressed CIE, the portion of the syllabus should not be common /repeated for any of the methods of the CIE. Each method of CIE should have a different syllabus portion of the course). CIE methods /test question paper is designed to attain the different levels of Bloom's taxonomy as per the outcome defined for the course.

##### **The sum of two tests, two assignments, will be out of 60 marks and will be scaled down to 30 marks CIE for the practical component of the Integrated Course**

- On completion of every experiment/program in the laboratory, the students shall be evaluated and marks shall be awarded on the same day. The **15 marks** are for conducting the experiment and preparation of the laboratory record, the other **05 marks shall be for the test** conducted at the end of the semester.
- The CIE marks awarded in the case of the Practical component shall be based on the continuous evaluation of the laboratory report. Each experiment report can be evaluated for 10 marks. Marks of all experiments' write-ups are added and **scaled down to 15 marks**.
- The laboratory test (**duration 02/03 hours**) at the end of the 14<sup>th</sup> /15<sup>th</sup> week of the semester /after completion of all the experiments (whichever is early) shall be conducted for 50 marks and **scaled down to 05 marks**.

Scaled-down marks of write-up evaluations and tests added will be CIE marks for the laboratory component of IPCC for **20 marks**.

## **Semester End Examination (SEE):**

### **SEE for IC**

Theory SEE will be conducted by University as per the scheduled time table, with common question papers for the course (duration 03 hours)

1. The question paper will have ten questions. Each question is set for 20 marks.
2. There will be 2 questions from each module. Each of the two questions under a module (with a maximum of 3 sub-questions), **should have a mix of topics** under that module.
3. The students have to answer 5 full questions, selecting one full question from each module.

**The theory portion of the Integrated Course shall be for both CIE and SEE, whereas the practical portion will have a CIE component only. Questions mentioned in the SEE paper shall include questions from the practical component).**

### **Passing standard:**

- The minimum marks to be secured in CIE to appear for SEE shall be 12 (40% of maximum marks-30) in the theory component and 08 (40% of maximum marks -20) in the practical component. The laboratory component of the IPCC shall be for CIE only. However, in SEE, the questions from the laboratory component shall be included. The maximum of 04/05 questions to be set from the practical component of IPCC, the total marks of all questions should not be more than 30 marks.
- SEE will be conducted for 100 marks and students shall secure 35% of the maximum marks to qualify for the SEE. Marks secured will be scaled down to 50.

## **Suggested Learning Resources:**

### **Textbooks**

1. Computer fundamentals and programming in c, “Reema Thareja”, Oxford University, Second edition, 2017.

### **Reference Books:**

1. E. Balaguruswamy, Programming in ANSI C, 7th Edition, Tata McGraw-Hill.
2. Brian W. Kernighan and Dennis M. Ritchie, The ‘C’ Programming Language, Prentice Hall of India.

### **Web links and Video Lectures (e-Resources):**

1. [elearning.vtu.ac.in/econtent/courses/video/BS/15PCD23.html](http://elearning.vtu.ac.in/econtent/courses/video/BS/15PCD23.html)
2. <https://nptel.ac.in/courses/106/105/106105171/> MOOC courses can be adopted for more clarity in understanding the topics and verities of problem solving methods.
3. <https://tinyurl.com/4xmrexre>

### **Activity Based Learning (Suggested Activities in Class)/ Practical Based learning**

- Quizzes
- Assignments
- Seminars

## CO-PO MAPPING

### COURSE OBJECTIVES:

This course (22POP13) will enable the students to:

1. Elucidate the basic architecture and functionalities of a Computer
2. Apply programming constructs of C language to solve the real-world problems
3. Explore user-defined data structures like arrays, structures and pointers in implementing solutions to problems
4. Design and Develop Solutions to problems using modular programming constructs such as functions and procedures.

### COURSE OUTCOMES:

At the end of the course the student will be able to

<b>CO1</b>	Elucidate the basic architecture and functionalities of a computer and also recognize the hardware parts.
<b>CO2</b>	Apply programming constructs of C language to solve the real-world problem
<b>CO3</b>	Explore user-defined data structures like arrays in implementing solutions to problems like searching and sorting
<b>CO4</b>	Explore user-defined data structures like structures, unions and pointers in implementing solutions
<b>CO5</b>	Design and Develop Solutions to problems using modular programming constructs

### PSOs

<b>PSO1</b>	Ability to adapt to a rapidly changing environment by learning and employing new programming skills and technologies
<b>PSO2</b>	Ability to use diverse knowledge across the domains with inter-personnel skills to deliver the Industry need

### MAPPING

CO/PO	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PO 12	PSO1	PSO2
<b>CO1</b>	2	1	1											
<b>CO2</b>	2	2	1									1	1	
<b>CO3</b>	3	2	2									2		1
<b>CO4</b>	2	2	2									2	1	1
<b>CO5</b>														

### CO-PO MAPPING JUSTIFICATION

CO-PO	LEVEL	JUSTIFICATION
CO1-PO1	2	The basic science/mathematical knowledge is required to be used to understand the concepts of computer.
CO1-PO2	1	The students should be able to identify and analyses the problems in the systems
CO1-PO3	1	The students should know the applications and need of systems for the society, health and safety.
CO2-PO1	2	The basic mathematical knowledge is needed about numbering system to understand the principles of c language like variable, arrays, structures etc...
CO2-PO2	2	The basic principles of mathematics/ science should be used to formulate solution to the given problem using C language.
CO2-PO3	1	The solution will be designed for the given problem using algorithm/ pseudo code/flowcharts
CO2-PO12	1	The fundamentals of C programming can be used in other programming languages also in the further semesters in the learning process
CO3-PO1	3	It meets highly, to give solution for the problem using C language need the fundamentals of mathematics and C language principles.
CO3-PO2	2	The solution given to the problem written in C is analyzed for the problems and complexity to have efficient program.
CO3-PO3	2	Solution for the complex problems are developed using C language slightly to the basics.
CO3-PO12	2	It meets highly, to give solution for the problem using C language need the fundamentals of mathematics and C language principles.
CO4-PO1	2	To solve the problems using modular approach like functions and structures should have the complete fundamental knowledge of C.
CO4-PO2	2	As we write programs using the functions and structures, used to analyze the problems with the previous methods that we referred and learn to use different methods.
CO4-PO3	2	Using the modular approaches like arrays, functions and structures we write the programs to solve with moderate collection of data and output to be produced.
CO4-PO12	2	The using of functions, arrays and user defined data types is life long process to deal with complex problems and huge data collection.
(CO2-CO4)-PSO1	2	With C basics the students able to adopt them to learn the other application programming languages. And they should write algorithms for the same.
(CO3, CO4)-PSO2	1	The C fundamentals of identifying and correcting logic and syntax errors can use with other domains.



## LAB ASSESSMENT RUBRICS FOR CIE

### 1. Maximum CIE: 20 Marks

- a. Continuous Evaluation of Experiments: 15 Marks
- b. Internal Test: 5 Marks

### 2. Rubrics for Continuous Evaluation of Experiments

Continuous Evaluation for 15 marks		Marks Allotted
A	Observation Write up and Attendance	4
B	Conduction of Experiment and Output	3
C	Viva Voice	4
D	Record Write up	4
Total		15

### 3. Rubrics for Test Evaluation: The laboratory test shall be conducted for 50 marks and scaled down to 05 marks.

Test Evaluation for 50 marks		Marks Allotted
A	Program Write-up	15
B	Conduction of Experiment and Output	20
C	Viva Voice	15
Total		50

## 4. Introduction to Principles of Programming Using C

**Problem Solving Techniques:** The process of working through details of a problem to reach a solution.

There are three approaches to problem solving:

- Algorithm
- Flowchart
- Pseudo Code

**Algorithm:** The algorithm is a step-by-step procedure to be followed in solving a problem. It provides a scheme to solve a particular problem in finite number of unambiguous steps. It helps in implementing the solution of a problem using any of the programming languages.

In order to qualify as an algorithm, a sequence of instructions must possess the following characteristics:

**Input:** It may accept a zero or more inputs.

**Output:** It should produce at least one output (result).

**Definiteness:** Each instruction must be clear, well defined and precise. There should not be any ambiguity.

**Finiteness:** It should be a sequence of finite instructions. That is, it should end after a fixed time. It should not enter into an infinite loop.

**Effectiveness:** This means that operations must be simple and are carried out in a finite time at one or more levels of complexity. It should be effective whenever traced manually for the results.

**Key features of an algorithm:** Any algorithm has a finite number of steps and some steps may involve decision making, repetition. Broadly speaking, an algorithm exhibits three key features that can be given as:

**Sequence:** Sequence means that each step of the algorithm is executed in the specified order.

**Decision:** Decision statements are used when the outcome of the process depends on some condition.

**Repetition:** Repetition which involves executing one or more steps for a number of times can be implemented using constructs like the while, do-while and for loops. These loops executed one or more steps until some condition is true.

### Example: To compute the Area of Rectangle

**Algorithm: Area\_of\_rectangle** [This algorithm takes length and breadth, the sides of the rectangle as input and computes the area of rectangle using the formula  $area = length * breadth$ . Finally, it prints the area of rectangle]

STEPS:

Step 1:[Initialize], Start



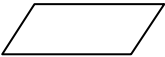
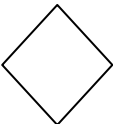




Step 2: [Input the sides of Rectangle], Read length, breadth

Step 3:[Compute the area of rectangle],  $Area = length * breadth$

Step 4:[Display the Area], Print Area

Step 5: [Finished], Stop

**Flowcharts:** A flowchart is a graphical or symbolic representation of an algorithm. They are basically used to design and develop complex programs to help the users to visualize the logic of the program so that they can gain a better understanding of the program and find flaws, bottlenecks, and other less-obvious features within it. Basically, a flowchart depicts the “*flow*” of a program. The following table shows the symbols used in flow chart along with its descriptions.

Symbol	Name	Description
	Oval	Represents the terminal point
	Rectangle	Represents the process steps defined in algorithm
	Parallelogram	Indicate the reading Operation used for input/output or data or information from/to any device
	Diamond	Indicates the decisions(questions) and consequently the Branch points or the paths to be followed based on the result of the question
	Arrows	Shows the flow chart direction and connects the various flow chart symbols
	Small circle	Shows the continuation from one point in the process flow To another
	Hexagon	Represents Looping structures
	Predefined Process	Indicates Subroutines

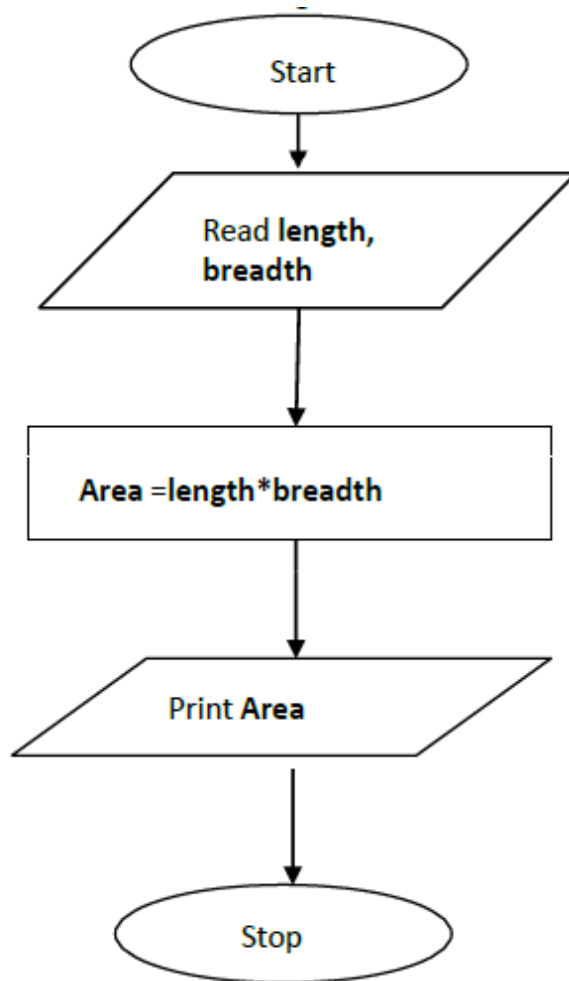
### Advantages of Flowcharts:

A flowchart is a diagrammatic representation that illustrates the sequence of steps that must be performed to solve a problem. They are usually drawn in the early stages of formulating computer solutions to facilitate communication between programmers and businesspeople.

Flowcharts help programmers to understand the logic of complicated and lengthy problems and they help to analyses the problem in a more effective manner.

Flow chart can be used to debug programs that have error(s).

Example: To compute the Area of Rectangle



## 5. Program - 1

### Simulation of Simple Calculator

**Program:**

```
#include <stdio.h>
void main( )
{
    char Operator;
    float num1, num2, result = 0;
    printf("Enter an Operator (+, -, *, /) :\n ");
    scanf("%c", &Operator);
    printf("Enter the Values for two Operands: num1 and num2 : \n ");
    scanf("%f%f", &num1, &num2);
    switch(Operator)
    {
        case '+':
            result = num1 + num2;
            printf("\n The result of %f %c %f = %f", num1, Operator, num2, result);
            break;
        case '-':
            result = num1 - num2;
            printf("\n The result of %f %c %f = %f", num1, Operator, num2, result);
            break;
        case '*':
            result = num1 * num2;
            printf("\n The result of %f %c %f = %f", num1, Operator, num2, result);
            break;
        case '/':
            result = num1 / num2;
            printf("\n The result of %f %c %f = %f", num1, Operator, num2, result);
            break;
        default: printf("\n You have entered an Invalid Operator ");
    }
}
```

**Test Cases:**

Test No	Input Parameters	Expected Output	Obtained Output	Remarks	Student Remarks
1	Operator=+ num1=9, num2=10	result=19	result=19	PASS	
2	Operator=/ num1=9, num2=0	Divide by zero error	Infinity	PASS	
3	Operator=/ num1=8, num2=2	result=4	result=4	PASS	
4	Operator=# num1=9, num2=10	Invalid operator	You have entered Invalid Operator	PASS	
<b>Test for the following cases and record your observation</b>					
5	Operator = *num1=5, num2=3				

## 6. Program –2

**Compute the roots of a quadratic equation by accepting the coefficients. Print appropriate messages.**

**Program:**

```
#include<stdio.h>
#include<math.h>
void main( )
{
float a,b,c,d,real,img,root1,root2;
printf("\n Enter the values for a, b and c\n");
scanf("%f%f%f",&a,&b,&c);
if(a==0)
{
printf("\n Invalid Co-efficient roots cannot be computed\n");
}
else
{
d=b*b-4*a*c;
printf("\n The value of Discriminant(d) is %f", d);
if(d==0)
{
printf("\n The real and equal roots are:\n");
root1 = root2 =-b/(2*a);
printf("root1=root2=%.2f",root1);
}
else if(d>0)
{
printf("\n The real and distinct roots are:\n");
root1=(-b+sqrt(d))/(2*a);
root2=(-b - sqrt(d))/(2*a);
printf("root1=%.2f and root2=%.2f",root1,root2);
}
else
{
printf("\n The complex roots are:\n");
real =-b/(2*a);
img=sqrt(fabs(d))/(2*a);
printf("root1=%.2f+%.2fi\t",real,img);
printf("root2 =%f-%.2fi",real,img);
}
}
}
```

**Test Cases:**

Test No	Input Parameters	Expected output	Obtained output	Remarks	Student remarks
1	a=0, b=0, c=0	Invalid coefficients Try Again with valid inputs !!!!	Invalid coefficients Roots cannot be computed	PASS	
2	a=1, b=6, c=9	Roots are real and equal Root1=-3and Root2=-3	Roots are real and equal Root1 = -3.000 and Root2=-3.000	PASS	
3	a=1, b=-5, c=3	Roots are real and distinct Root1= 4.30 and Root2=0.69	Roots are real and distinct Root1 = 4.303 and Root2=0.697	PASS	
4	a=1, b=4, c=7	Roots are real and imaginary Root1=-2+i1.73 Root2=-2-i1.73	Roots are real and imaginary Root1=2.000+i1.732 Root2=-2.000-i1.732	PASS	
<b>Test for the following cases &amp; record your observations</b>					
5	a=0, b=4, c=5				
6	a=2, b=-7, c=0				
7	a=2, b=-7, c=8				



## 7. Program - 3

An electricity board charges the following rates for the use of electricity: for the first 200 units 80 paise per unit: for the next 100 units 90 paise per unit: beyond 300 units Rs 1 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 name of the user, number of units consumed and print out the charges.

**Program:**

```
#include<stdio.h>
void main( )
{
    char uname[15];
    int units;
    float bill, penalty, Total_Amount;
    printf("enter the username\n");
    scanf("%s",uname);
    printf("enter the number of units consumed\n");
    scanf("%d",&units);
    if(units<=200)
    {
        bill=units*0.80+100;
    }
    else if(units>=201&&units<=300)
    {
        bill=(200*0.80)+((units-200)*0.90)+100;
    }
    else
    {
        bill=(200*0.80)+(100*0.90)+((units - 300)*1.00);
    }
    if(bill>400)
    {
        penalty=bill*0.15;
    }
    Total_Amount = bill+penalty;
    printf("\n Electricity Bill Details of Customer Are:\n");
    printf(" Customer name is:%s",uname);
    printf(" Total number of Units Consumed are: %d", units);
    printf(" Total Amount to be payed is: %f", Total_Amount);
}
```

**Test Cases:**

Test No	Input Parameters	Expected output	Obtained output	Remarks	Student Remarks
1	n=100	Total Amount to be paid=180Rs	Total Amount to be paid=180Rs	PASS	
2	n=250	Total Amount to be paid=305Rs	Total Amount to be paid=305Rs	PASS	
3	n=400	Total Amount to be paid=517.5Rs	Total Amount to be paid=517.5Rs	PASS	
<b>Test for the following cases &amp; record your observations</b>					
4	n=500				
5	n=1000				
6	n=770				

## 8. Program – 4

Write a C Program to display the following by reading the number of rows as input,

```
    1
   1 2 1
  1 2 3 2 1
 1 2 3 4 3 2 1
-----
nth row
```

**Program:**

```
#include <stdio.h>
void main()
{
    int i,j,n;
    printf("Input number of rows : ");
    scanf("%d",&n);
    for(i=0;i<=n;i++)
    {
        /* print blank spaces */
        for(j=1;j<=n-i;j++)
            printf(" ");
        /* Display number in ascending order upto middle*/
        for(j=1;j<=i;j++)
            printf("%d",j);

        /* Display number in reverse order after middle */
        for(j=i-1;j>=1;j--)
            printf("%d",j);
        printf("\n");
    }
}
```

**Test Cases:**

Test No	Input Parameters	Expected output	Obtained output	Remarks	Student Remarks
1	Input number of rows n=3	<pre> 1 1 2 1 1 2 3 2 1 </pre>	<pre> 1 1 2 1 1 2 3 2 1 </pre>	PASS	
2	Input number of rows n=4	<pre> 1 1 2 1 1 2 3 2 1 1 2 3 4 3 2 1 </pre>	<pre> 1 1 2 1 1 2 3 2 1 1 2 3 4 3 2 1 </pre>	PASS	
<b>Test for the following cases &amp; record your observations</b>					
4	n=5				
5	n=6				

## 9. Program – 5

### Implement Binary Search on Integers

#### Program:

```
#include<stdio.h>
void main()
{
    int n,i,low,high,mid,flag=0;
    int key,a[50];
    printf("\n Enter the number of elements:\n");
    scanf("%d",&n);
    printf("\nEnter the elements of array :\n");
    for(i=0;i<n;i++)
    {
        scanf("%d",&a[i]);
    }
    printf("\nEnter the key to be searched:\n");
    scanf("%d",&key);
    low=0;
    high=n-1;
    while(low<=high)
    {
        mid=(low+high)/2;
        if(key==a[mid])
        {
            printf("\nkey is found at location %d\n",mid+1);
            flag=1;
            break;
        }
        else if(key>a[mid])
        {
            low=mid+1;
        }
        else
        {
            high=mid-1;
        }
    }
    if(flag==0)
    {
        printf("\n key not found in the list \n");
    }
}
```

**Test Cases:**

Test no	Input Parameters	Expected output	Obtained output	Remarks	Student Remarks
1	n=5 a[0]=23 a[1]=74 a[2]=89 a[3]=65 a[4]=39 key=65	Element not found in the List	Element not found in the List	PASS	
2	n=5 a[0]=10 a[1]=20 a[2]=37 a[3]=40 a[4]=77 key=20	Element found at position: 2	Element found at position: 2	PASS	
<b>Test for the following cases &amp; record your observations</b>					
3	n=5 a[0]=28 a[1]=100 a[2]=999 a[3]=566 a[4]=456 key=566				

## 10. Program – 6

**Implement Matrix multiplication and validate the rules of multiplication**

**Program:**

```
#include<stdio.h>
void main()
{
int a[10][10],b[10][10],c[10][10];
int m,n,p,q,i,j,k;
printf("\n enter the order of the matrix A\n");
scanf("%d%d",&m,&n);
printf("\n enter the order of the matrix B\n");
scanf("%d%d",&p,&q);
if(n!=p)
{
printf("\n Invalid inputs, Multiplication is not Possible \n");
}
else
{
printf("\n enter the elements of matrix A\n");
for(i=0;i<m;i++)
{
for(j=0;j<n;j++)
{
scanf("%d",&a[i][j]);
}
}
printf("\n enter the elements of matrix B\n");
for(i=0;i<p;i++)
{
for(j=0;j<q;j++)
{
scanf("%d",&b[i][j]);
}
}
/* Multiplication */
for(i=0;i<m;i++)
{
for(j=0;j<q;j++)
{
c[i][j]=0;
for(k=0;k<p;k++)
{
c[i][j] = c[i][j]+a[i][k]*b[k][j];
}
}
}
printf("\n The Resultant Matrix is: \n");
for(i=0;i<m;i++)
{
```

```

for(j=0;j<n;j++)
{
printf("%d\t",c[i][j]);
}
printf("\n");
}
}
}

```

**Test Cases:**

Test No	Input Parameters	Expected output	Obtained output	Remarks	Student Remarks
1	Matrix A Size=2,2 Matrix B Size= 2, 2Elements= 12 56 34 * 7 8	The product of two matrix 19 22 43 50	The product of two matrix 19 22 43 50	PASS	
2	Matrix A Size=2,3 Matrix B Size=2,2	Invalid input	Multiplication not possible	PASS	
<b>Test for the following cases &amp; record your observations.</b>					
5	Matrix A Size=2,3 Matrix B Size=3, 2				



## 11. Program – 7

**Compute  $\sin(x)/\cos(x)$  using Taylor series approximation. Compare your result with the built-in library function. Print both the results with appropriate inferences.**

**Program:**

```
#include<stdio.h>
#include<math.h>
#define PI 3.142
int main()
{
    int i, degree;
    float x, sum=0,term,nume,deno;
    printf("Enter the value of degree");
    scanf("%d",&degree);
    x = degree * (PI/180); //converting degree into radian
    nume = x;
    deno = 1;
    i=2;
    do
    {
        term = nume/deno;
        nume = -nume*x*x;
        deno = deno*i*(i+1);
        sum=sum+term;
        i=i+2;
    } while (fabs(term) >= 0.00001); // Accurate to 4 digits
    printf("The computed sin(x) value of %d is %.3f\n", degree, sum);
    printf("Using Built function sin(x) value of %d is %.3f", degree, sin(x));
    return 0;
}
```

**Test Cases:**

Test No	Input Parameters	Expected output	Obtained output	Remarks	Student Remarks
1	degree=0	The sine of 0 is =0	The sine of 0 is=0.000 Using in built function sin (0) =0.000	PASS	
2	degree=30	The sine of 30=0.500	The sine of 30 is=0.500 Using inbuilt function sin (30)=0.500	PASS	
3	degree=60	The sine of 60=0.866	The sine of 60is=0.866 Using inbuilt function sin ( 60 )=0.866	PASS	
4	degree = -10	The sine of -10= -0.17	The sine of -10 is= - 0.174 Using inbuilt function sin (-10)=-0.174	PASS	
<b>Test for the following cases &amp; record your observations.</b>					
5	degree=90				
6	degree=45				

**12. Program –8****Sort the given set of N numbers using Bubble sort****Program:**

```

#include<stdio.h>
void main( )
{
int n,i,j,temp,a[100];
printf("\n Enter the value for n:\n");
scanf("%d",&n);
printf("\n Enter %d elements into array:\n",n);
for(i=0;i<n;i++)
{
scanf("%d",&a[i]);
}
printf("\n The original elements are:\n");
for(i=0;i<n;i++)
{
printf("%d\t",a[i]);
}
/* Sorting */
for(i=1;i<n-1;i++)
{
for(j=0 ; j< n-i-1 ; j++)
{

```

```
if(a[j]>a[j+1])
{
temp=a[j];
a[j]=a[j+1];
a[j+1]=temp;
}
}
}
printf("\nThe sorted array is:\n");
for(i=0;i<n;i++)
{
printf("%d\t",a[i]);
}
}
```

**Test Cases:**

Test No	Input Parameters	Expected output	Obtained output	Remarks	Student Remarks
1	n=5 a[5]={23,4,6,12,40}	Sorted ele are: a[5]={4,6,12,23,40}	a={23,4,6,12,40} Sorted a={4,6,12,23,40}	PASS	
2	n=4 a[4]={67,5,0,34}	Sorted ele are: a[4]={0,5,34,67}	a={67,5,0,34} Sorted a={0,5,34,67}	PASS	
3	n=4 a[4]={-3,-9,12,6}	Sorted ele are: a[4]={-9,-3,6,12}	a={-3,-9,12,6} Sorted a={-9,-3,6,12}	PASS	
4	n=5 a[5]={12,23,6,5,4,6}	Sorted ele are: a[5]={6,6,12,23,54}	a={12,23,6,54,6} Sorted a={6,6,12,23,54}	PASS	
<b>Test for the following cases &amp; record your observations</b>					
5	n=8				
6	n=15				

### 13. Program – 9

**Write functions to implement string operations such as compare, concatenate, and find string length. Use the parameter passing techniques**

**Program:**

```
#include<stdio.h>
#include<string.h>
void compare(char [ ],char [ ]);
void concat(char [ ],char [ ]);
void length(char *[ ]);
void main( )
{
    int n,digit;
    char str1[10],str2[10];
    do
    {
        printf("press 1-compare 2-concatenate 3-length of string");
        printf("\n enter your choice=");
        scanf("%d",&n);
        switch(n)
        {
            case 1:printf("enter first string=");
                scanf("%s",str1);
                printf("enter second string=");
                scanf("%s",str2);
                compare(str1,str2);
                break;
            case 2: printf("enter first string=");
                scanf("%s",str1);
                printf("enter second string=");
                scanf("%s",str2);
                concat(str1,str2);
                break;
            case 3:printf("enter string=");
                scanf("%s",str1);
                length(&str1);
                break;
            default: printf("wrong choice");
                break;
        }
        printf("\n Do you want to continue(1/0)? ");
        scanf("%d", &digit);
    }
    while(digit==1);
}
void compare(char str1[ ],char str2[ ])
{
    int i;
```

```
i=strcmp(str1,str2);
if(i==0)
{
printf("strings are equal\n ");
}
else
{
printf("string are not equal\n");
}
}
void concat(char str1[ ],char str2[ ])
{
strcat(str1,str2);
printf("concatenate string=%s",str1);
}
void length(char *str1[ ])
{
int len;
len=strlen(str1);
printf("the length of string=%d",len);
}
```

**Test Cases:**

Test No	Input Parameters	Expected output	Obtained output	Remarks	Student Remarks
1	s1=NCET s2= CSE	Length of the string s1 is :3 Strings are not equivalent Concatenated String is: NCETCSE	Length of the string s1 is : 4Strings are not equivalent Concatenated string is: NCETCSE	PASS	
2	s1 = DATA s2=TYPE	Length of the string s1 is : 4Strings are not equivalent Concatenated string is: DATATYPE	Lengthofthestrings1is:4Strings are not equivalent Concatenated string is : DATATYPE	PASS	
3	s1 = CSE s2=CSE	Lengthofthestrings1is : 3 Both the strings are equal Concatenated String is: CSECSE	Length of the string s1 is : 3Both the strings are equal Concatenated string is: CSECSE	PASS	
<b>Test for the following cases &amp;record your observations.</b>					
4.	s1 = COMPUTER s2 = SCIENCE				

## 14. Program - 10

**Implement structures to read, write and compute average- marks of the students, list the students scoring above and below the average marks for a class of N students**

**Program:**

```
#include <stdio.h>
struct student
{
char usn[50];
char name[50];
int marks;
} s[10];
void main()
{
int i,n,countav=0,countbv=0;
float sum,average;
printf("\n Enter number of Students\n");
scanf("%d",&n);
printf("\n Enter information of students:\n");
// storing information
for(i=0; i<n;i++)
{
printf("\n Enter USN of student %d: ",i+1);
scanf("%s",s[i].usn);
printf("\n Enter name of student %d: ",i+1);
scanf("%s",s[i].name);
printf("\n Enter marks of student %d: ",i+1);
scanf("%d",&s[i].marks);
printf("\n");
}
printf("Displaying Information:\n\n");
// displaying information
for(i=0; i<n; i++)
{
printf("\nUSN of student %d is : %s\n",i+1,s[i].usn);
printf("Name of student %d is : ",i+1);
puts(s[i].name);
printf("Marks of student %d is : %d",i+1, s[i].marks);
printf("\n");
}
for(i=0,i<n;i++)
{
sum=sum+s[i].marks;
}
average=sum/n;
printf("\nAverage marks is : %f",average);
countav=0;
```



```
countbv=0;
for(i=0;i<n;i++)
{
if(s[i].marks>=average)
countav++;
else
countbv++;
}
printf("\nTotal No of students scored above average= %d",countav);
printf("\nTotal No of students scored below average= %d",countbv);
}
```

## Test Cases:

Test No	Input Parameters				Expected output	Obtained output	Remarks	Student Remarks
1	n=3, Enter the student details				Averagemarks:520 Total No of students above average=1 Total No of students below average=2	Averagemarks:520 Total No of students above average=1 Total No of students below average=2	PASS	
		usn	name	marks				
	s[0]	NC18CS001	Chetan	500				
	s[1]	NC18CS002	Darshan	510				
	s[2]	NC18CS003	Pallavi	550				
2	n=2, Enter the student details				Average marks:540 Total No of students above average=1 Total No of students below average=1	Averagemarks:540 Total No of students above average=1 Total No of students below average=1	PASS	
		usn	name	marks				
	s[0]	NC18IS001	Rajesh	520				
	s[1]	NC18IS002	Darshan	560				
Test for the following cases &record your observations								
3	n=2, Enter the student details							
		usn	name	marks				
	s[0]	NC18IS009	Rama	520				
	s[1]	NC18IS006	Sita	560				

## 15. Program – 11

**Develop a program using pointers to compute the sum, mean and standard deviation of all elements stored in an array of N real numbers**

**Program:**

```
#include<stdio.h>
#include<math.h>
int main()
{
float a[10], *ptr, mean, std, sum=0, sumstd=0;
int n,i;
printf("\n Enter the no of elements\n");
scanf("%d",&n);
printf("\n Enter the array elements\n");
for(i=0;i<n;i++)
{
scanf("%f",&a[i]);
}
ptr=a;
for(i=0;i<n;i++)
{
sum=sum+ *ptr;
ptr++;
}
mean=sum/n;
ptr=a;
for(i=0;i<n;i++)
{
sumstd=sumstd + pow((*ptr - mean),2);
ptr++;
}
std= sqrt(sumstd/n);
printf("\n Sum=%.3f\t",sum);
printf("\n Mean=%.3f\t",mean);
printf("\n Standard deviation=%.3f\t",std);
return 0;
}
```

**Test Cases:**

Test No	Input Parameters	Expected output	Obtained output	Remarks	Student Remarks
1	n=5 Array elements 1 5 9 6 7	sum=28 mean=5.6 std=2.653	sum=28.000 mean=5.600 std=2.653	PASS	
2	n=4 Array elements 2.3 1.1 4.5 2.7 8	sum=10.68 mean=2.67 std=1.22	sum=10.680 mean=2.670 std=1.221	PASS	
<b>Test for the following cases &amp; records your Observations</b>					
3	n=5 Array elements 2 3 4 8 10				
4	n=6 Array elements 2.4 5.6 2.0 4.12 5.17 0.14				

## 16. Program -12

**Write a C program to copy a text file to another, read both the input file name and target file name.**

**Program:**

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

void main()
{
    FILE *fptr1, *fptr2;
    char ch, fname1[20], fname2[20];

    printf("\n\n Copy a file in another name :\n");
    printf(".....\n");

    printf(" Input the source file name : ");
    scanf("%s",fname1);

    fptr1=fopen(fname1, "r");
    if(fptr1==NULL)
    {
        printf(" File does not found or error in opening.!!");
        exit(1);
    }
    printf(" Input the new file name : ");
    scanf("%s",fname2);
    fptr2=fopen(fname2, "w");
    if(fptr2==NULL)
    {
        printf(" File does not found or error in opening.!!");
        fclose(fptr1);
        exit(2);
    }
    while(1)
    {
        ch=fgetc(fptr1);
        if(ch==EOF)
        {
            break;
        }
        else
        {
            fputc(ch, fptr2);
        }
    }
    printf(" The file %s copied successfully in the file %s. \n\n",fname1,fname2);
    fptr1=fopen(fname1, "r");
    fptr2=fopen(fname2, "r");
```

```

    fclose(fp1);
    fclose(fp2);
    getchar();
}

```

**Test Cases:**

Test No	Input Parameters	Expected output	Obtained output	Remarks	Student Remarks
1	Input File name: Test.txt  Target File name: Test1.txt	File test1.txt is copied successfully to file test2.txt	File test1.txt is copied successfully to file test2.txt	PASS	
2	Input File name: Test4.txt  Target File name: Test15.txt	Input file does not find	Input file does not find	PASS	
<b>Test for the different cases and records your observations</b>					
3					
4					

## Viva Voice Questions

1. What is an algorithm?
2. What are the characteristics of an algorithm?
3. What are the notations used while writing an algorithm?
4. What is flowchart?
5. List the symbols used while writing flowchart.
6. What is pseudocode?
7. What is ASCII?
8. What is high level language?
9. What is compiler?
10. What are tokens?
11. What are keywords? How many keywords are there in C programming language?
12. What is a variable?
13. What is the significance of a variable?
14. What are the rules to be followed while declaring a variable?
15. What is a constant?
16. What is a datatype? What are the different data types?
17. What are the basic or primary or fundamental datatypes supported by C?
18. What are escape sequence characters?
19. What are backslash constants? Name some constants.
20. List the size and range of basic data types.
21. What is the difference between a character and string containing a single character?
22. What is the meaning of associativity of an operator?
23. What is left associativity and right associativity?
24. What is implicit type conversion and explicit type conversion (type casting)?
25. What is precedence of an operator means?
26. List the precedence of all the types of operators along with associativity.
27. List the formatted input and output functions.
28. What is an expression?
29. What are the different types of expressions?
30. What is function?
31. What are the advantages of functions?
32. What are the different types of functions?
33. What are the elements of functions?
34. What is a library function?
35. What is calling function and called function?
36. What is the meaning of actual parameter and formal parameter?
37. What is function prototype or function declaration?
38. What is a function call?
39. Which are the logical operators?
40. Define decision making statement?
41. What are the verities of if-statements?
42. What is the purpose of switch statement? Explain with syntax
43. What is loop? List the differences between pre-test and post-test loop
44. What are the advantages of loops?

45. What is control statement? What are the various types of control statements available in C
46. Explain for loop with syntax.
47. What is the difference between while and do-while loop?
48. What are unconditional control statements?
49. What is the use of break statement?
50. What is recursion? What are the advantages and disadvantages of recursion?
51. What is the difference between an ordinary variable and an array variable?
52. How an array will be initialized? Explain with an example.
53. How to declare an array variable?
54. What is bubble sort?
55. What is binary search?
56. What is multi-dimensional array?
57. What are the differences between recursion and iteration?
58. What is string? How strings are represented?
59. How the strings are stored in memory?
60. What is the difference between a character and string containing a single character?