CMR INSTITUTE OF TECHNOLOGY

Department of MCA

Internal Assessment Test I – Dec 2025

Sub:	Programming and Problem Solving in C							**Sub Code:**	MMC101
Date:	**5-02-2025**	**Duration:**	**90 min's**	**Max Marks:**	**50**	**Sem:**	**I**	**Branch:**	**MCA**

Note : Answer FIVE FULL Questions, choosing ONE full question from each Module

	PART I	MARKS	OBE	
			CO	RBT
1	Give the structure of C program. Also explain the compilation process in detail. **OR**	[10]	CO1	L1
2	Explain different categories of conditional statements with syntax and examples	[10]	CO1	L1
	PART II			
3	Explain the characteristics of the C language and its applications **OR**	[10]	CO1	L2
4	Explain the different types of looping statements in C with examples. Also give the use of break and continue statements in loops.	[10]	CO1	L2
	PART III			
5	Write an algorithm and develop a C program that reads N integer numbers and arrange them in ascending order using Bubble Sort. **OR**	[10]	CO2	L3
6	What is an array? Write a C program to input N integers and find largest element in array.	[10]	CO2	L1,L3
	PART IV			
7	How to declare and initialize a 2-D array? Write a program to add 2 matrices. **OR**	[10]	CO2	L3
8	Define String. Write a C program to count the number of letters, digits, and special characters in a string without using built-in functions	[10]	CO2	L2
	PARTV			
9	Write short notes on: a. Ternary operator b. Enumeration c. Type casting **OR**	[10]	CO1	L1,L3
10	Write a menu-driven C program to perform basic operations on a 1-D array	[10]	CO2	L4

1. **Give the structure of C program. Also explain the compilation process in detail.**

```
Documentation Section
Pre-processor directives
Definition Section and Global declarations void
main()
{
        Declaration part

        Executable part

}
Sub Program Section
{
        Body of the Sub

}
```

Comments:

- Comments are used to convey a message and used to increase the readability of a program.
- They are not processed by the compiler. There are two types of comments:
    1. Single line comment
    2. Multi line comment

Single line comment

A single line comment starts with two forward slashes (//) and is automatically terminated with the end of the line.

E.g. //First C program

Multi-line comment

A multi-line comment starts with /* and terminates with */.

E.g. /* This program is used
to find Area of the Circle */

Section 1.Preprocessor Directive section

- The preprocessor directive section is optional.
- The pre-processor statement begins with # symbol and is also called the pre-processor directive.
- These statements instruct the compiler to include C preprocessors such as header files and symbolic constants before compiling the C program.
- They are executed before the compiler compiles the source

code. Some of the pre-processor statements are listed below:

(i)Header files

#include<stdio.h>- to be included to use standard I/O functions : prinf(),scanf()

(ii)Symbolic constants

#define PI 3.1412

#define TRUE 1

Section 2: Global declaration Section

This section is used to declare a global variable. These variables are declared before the main() function as well as user defined functions.

Global variables are variables that are used in more than one function.

Section 3: Function Section

This section is compulsory. This section can have one or more functions. Every program written in C language must contain main () function. The execution of every C program always begins with the function main ().

Every function consists of 2 parts

1. Header of the function
2. Body of the function

2. Explain different categories of conditional statements with syntax and examples

1. **if Statement -** The if statement is used to execute a block of code only when a given condition is true.

```
if (a > b)
   printf("a is greater");
```

2. **if–else Statement -** The if–else statement executes one block of code when the condition is true and another block when the condition is false.

```
if (a % 2 == 0)
   printf("Even");
else
   printf("Odd");
```

3. **else–if Ladder -** The else–if ladder is used to test multiple conditions sequentially and executes the block corresponding to the first true condition.

```
if (marks >= 90)
   grade = 'A';
else if (marks >= 75)
   grade = 'B';
else
   grade = 'C';
```

4. **Nested if -** A nested if statement is an if statement placed inside another if statement to test multiple related conditions.

```
if (a > b) {
   if (a > c)
      printf("a is largest");
}
```

5. **switch Statement -** The switch statement allows multi-way selection by executing different code blocks based on the value of an expression.

```
switch(day) {
   case 1: printf("Monday"); break;
   default: printf("Invalid");
}
```

3. **Explain the characteristics of the C language and its applications**

Characteristics

- Simple and structured
- Portable
- Efficient and fast
- Supports pointers
- Rich library support

Applications

- Operating systems
- Embedded systems
- Compiler design
- Game development
- Database systems

4. **Explain the different types of looping statements in C with examples. Also give the use of break and continue statements in loops.**

In C programming, looping statements are used to execute a block of code multiple times until a specified condition is met. There are three types of loops:

1. for Loop

The for loop is used when the number of iterations is known beforehand. It consists of three parts: initialization, condition, and increment/decrement.

Syntax:

```
for(initialization; condition; increment/decrement) {
   // Code to be executed
}
```

Example:

```
#include <stdio.h>
int main() {
   for(int i = 1; i <= 5; i++) {
      printf("%d ", i);
   }
   return 0;
}
```

Output:
1 2 3 4 5

2. while Loop

The while loop is used when the number of iterations is not known in advance. It executes as long as the condition is true.

Syntax:

```
while(condition) {
   // Code to be executed
}
```

Example:

```
#include <stdio.h>
int main() {
   int i = 1;
```

```
    while(i <= 5) {
        printf("%d ", i);
        i++;
    }
    return 0;
}
```

Output:
1 2 3 4 5

3. do-while Loop

The do-while loop executes the block of code at least once, regardless of the condition.

Syntax:

```
do {
    // Code to be executed
} while(condition);
```

Example:

```
#include <stdio.h>
int main() {
    int i = 1;
    do {
        printf("%d ", i);
        i++;
    } while(i <= 5);
    return 0;
}
```

Output:
1 2 3 4 5

Use of break and continue Statements in Loops

1. break Statement

The break statement is used to exit a loop prematurely when a certain condition is met.

Example:

```
#include <stdio.h>
int main() {
    for(int i = 1; i <= 5; i++) {
        if(i == 3)
            break;  // Exits the loop when i is 3
        printf("%d ", i);
    }
    return 0;
}
```

Output:
1 2

2. continue Statement

The continue statement is used to skip the current iteration of the loop and move to the next iteration.

Example:

```
#include <stdio.h>
int main() {
   for(int i = 1; i <= 5; i++) {
      if(i == 3)
         continue;  // Skips iteration when i is 3
      printf("%d ", i);
   }
   return 0;
}
```

Output:
1 2 4 5

5. **Write an algorithm and develop a C program that reads N integer numbers and arrange them in ascending order using Bubble Sort.**

Algorithm for Bubble Sort

1. First, we will select the range of the unsorted array. For that, we will run a loop(say i) that will signify the last index of the selected range. The loop will run backward from index n-1 to 0(where n = size of the array). The value i = n-1 means the range is from 0 to n-1, and similarly, i = n-2 means the range is from 0 to n-2, and so on.
2. Within the loop, we will run another loop(say j, runs from 0 to i-1 though the range is from 0 to i) to push the maximum element to the last index of the selected range, by repeatedly swapping adjacent elements.
   Basically, we will swap adjacent elements(**if arr[j] > arr[j+1]**) until the maximum element of the range reaches the end.
3. Thus, after each iteration, the last part of the array will become sorted. Like: after the first iteration, the array up to the last index will be sorted, and after the second iteration, the array up to the second last index will be sorted, and so on.
4. After (n-1) iteration, the whole array will be sorted.

```
#include <stdio.h>
int main() {
   int a[10], n, i, j, temp;
   scanf("%d", &n);
   for(i=0;i<n;i++)
      scanf("%d",&a[i]);

   for(i=0;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;
         }
```

```
    for(i=0;i<n;i++)
        printf("%d ", a[i]);
    return 0;
}
```

6. **What is an array? Write a C program to input N integers and find largest element in array.**

An array in C is a collection of elements of the same data type stored at contiguous memory locations. It allows efficient access and manipulation of multiple values using a single variable name and index.

```
#include <stdio.h>
int main() {
    int a[10], n, i, max;
    scanf("%d", &n);
    for(i=0;i<n;i++)
        scanf("%d",&a[i]);

    max = a[0];
    for(i=1;i<n;i++)
        if(a[i] > max)
            max = a[i];

    printf("Largest = %d", max);
    return 0;
}
```

7. **How to declare and initialize a 2-D array? Write a program to add 2 matrices .**

Declaration of a 2-D Array

A two-dimensional array is declared by specifying the number of rows and columns.

Syntax:

data_type array_name[row_size][column_size];

Example:

int a[3][3];

Initialization of a 2-D Array

A 2-D array can be initialized at the time of declaration.

Example:

```
int a[2][2] = {
    {1, 2},
    {3, 4}
};
```

Program to Add 2 matrices

```
#include <stdio.h>

int main() {
    int a[10][10], b[10][10], c[10][10];
    int i, j, r, col;
```

```
    printf("Enter number of rows and columns: ");
    scanf("%d %d", &r, &col);

    printf("Enter elements of first matrix:\n");
    for(i = 0; i < r; i++) {
        for(j = 0; j < col; j++) {
            scanf("%d", &a[i][j]);
        }
    }

    printf("Enter elements of second matrix:\n");
    for(i = 0; i < r; i++) {
        for(j = 0; j < col; j++) {
            scanf("%d", &b[i][j]);
        }
    }

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

    printf("Sum of the matrices:\n");
    for(i = 0; i < r; i++) {
        for(j = 0; j < col; j++) {
            printf("%d ", c[i][j]);
        }
        printf("\n");
    }

    return 0;
}
```

8. **Define String. Write a C program to count the number of letters, digits, and special characters in a string without using built-in functions**.

A **string** in C is a sequence of characters stored in a character array and terminated by a null character '\0'. Strings are used to store and manipulate text such as names, words, and sentences.

Example:

```
    char str[20] = "Hello";
#include <stdio.h>

int main() {
    char str[100];
    int i;
    int letters = 0, digits = 0, special = 0;

    printf("Enter a string: ");
    gets(str);

    for(i = 0; str[i] != '\0'; i++) {
        if((str[i] >= 'A' && str[i] <= 'Z') ||
           (str[i] >= 'a' && str[i] <= 'z')) {
```

```
            letters++;
        }
        else if(str[i] >= '0' && str[i] <= '9') {
            digits++;
        }
        else {
            special++;
        }
    }

    printf("Letters: %d\n", letters);
    printf("Digits: %d\n", digits);
    printf("Special characters: %d\n", special);

    return 0;
}
```

9. **Write short notes on: a. Ternary operator b. Enumeration c. Type casting**

a) Ternary Operator

The **ternary operator** is a conditional operator used to evaluate an expression and return one of two values based on a condition.
It is also known as the **conditional operator** and is represented by ?:.

Syntax:

condition ? expression1 : expression2;

Example:

max = (a > b) ? a : b;

b) Enumeration

Enumeration (enum) is a user-defined data type in C that consists of a set of named integer constants.
It improves code readability and helps represent fixed sets of values.

Syntax:

enum days {Mon, Tue, Wed, Thu, Fri};

Example:

enum days d;
d = Mon;

c) Type Casting

Type casting is the process of converting a variable from one data type to another.
It is mainly used to avoid data loss or to get correct results in expressions.

Syntax:

(data_type) variable;

Example:

float avg = (float)sum / n;

10. **Write a menu-driven C program to perform basic operations on a 1-D array**.

```
#include <stdio.h>

int main() {
    int a[50], n, i, choice, pos, value, found;

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

    printf("Enter array elements:\n");
    for(i = 0; i < n; i++)
        scanf("%d", &a[i]);

    do {
        printf("\nMENU");
        printf("\n1. Insert an element");
        printf("\n2. Delete an element");
        printf("\n3. Search for an element");
        printf("\n4. Display the array");
        printf("\n5. Exit");
        printf("\nEnter your choice: ");
        scanf("%d", &choice);

        switch(choice) {

        case 1:
            printf("Enter position to insert: ");
            scanf("%d", &pos);
            printf("Enter value to insert: ");
            scanf("%d", &value);

            for(i = n; i > pos - 1; i--)
                a[i] = a[i - 1];

            a[pos - 1] = value;
            n++;
            break;

        case 2:
            printf("Enter position to delete: ");
            scanf("%d", &pos);

            for(i = pos - 1; i < n - 1; i++)
                a[i] = a[i + 1];

            n--;
            break;

        case 3:
            printf("Enter element to search: ");
            scanf("%d", &value);
            found = 0;

            for(i = 0; i < n; i++) {
                if(a[i] == value) {
                    printf("Element found at position %d\n", i + 1);
                    found = 1;
                    break;
                }
            }
```

```
            if(found == 0)
                printf("Element not found\n");
            break;

        case 4:
            printf("Array elements:\n");
            for(i = 0; i < n; i++)
                printf("%d ", a[i]);
            printf("\n");
            break;

        case 5:
            printf("Exiting program...\n");
            break;

        default:
            printf("Invalid choice\n");
        }

    } while(choice != 5);

    return 0;
}
```