Graphic Era
Deemed to be
University
DEHRADUN

Directorate of Distance and Online Education

Name of Student: Deepankar Sharma

Student ID: 233512013

Subject Code: OMC 103

Subject Name: Programming & Problem Solving

Programme Name: MCA

Semester: 01

Sharma

Signature of the Student

Name: Deepankar Sharma
Course: MCA
Student ID: 233512013

Ques① (a) Approaches used in computational thinking

① Decomposition : Breaking down a complex problem into smaller, more mangeable subproblems or tasks.

② Pattern Recognition : Identifying patterns and trends within data or problems to find similarity or regularities.

③ Abstraction : Focusing on essential details while ignoring irrelevant information to simplify problem solving.

④ Algorithm Design : Developing step by step instructions or procedures to solve a specific problem efficiently.

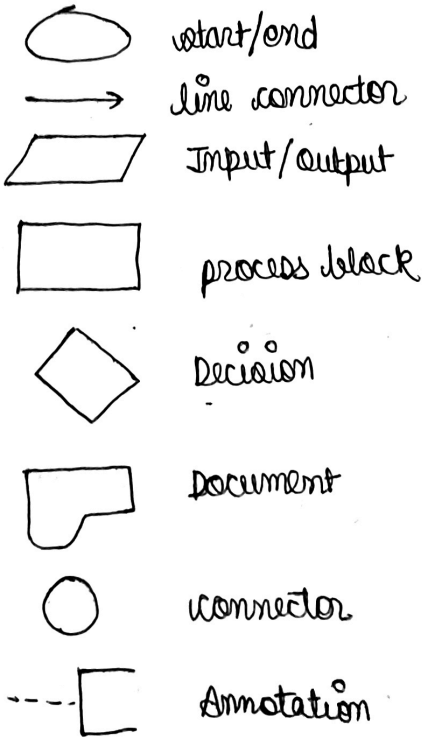
⑤ Algorithm Analysis : Evaluating the efficiency and effectiveness of algorithms in terms of time complexity, space complexity & other relevant metrics

⑥ Generalization : Applying solutions or patterns from one domain to solve similar problems in other domains.

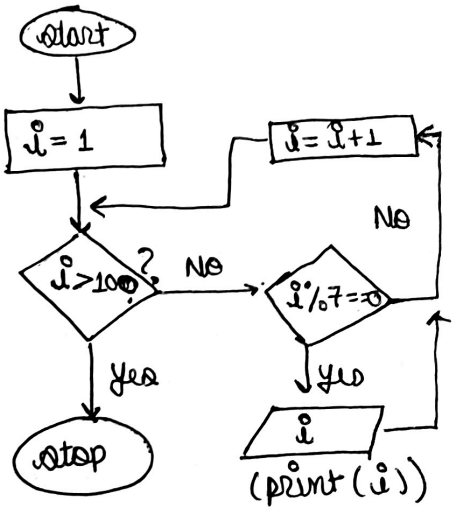
⑦ Evaluation : Accessing the correctness, completeness and effectiveness of solutions through testing and validation.

Flowcharts

Following are the symbols used in flowcharts:



Flowchart for multiples of 7



Name: Deepankar Sharma
Course: MCA
Student ID: 23351 2013

Ques ② (a) **Rules for Naming variables in C**

→ Variable names must begin with alphabet or an underscore
→ Variable names can contain alphabets, numbers (0-9) & underscores (_).
→ C is case sensitive, so variable "ram", "Ram", "rAM" are all different.
→ Variable names cannot be the names of keyword.
→ conventions regarding naming variables is they should be descriptive & meaningful, reflecting the purpose or meaning of their usage.

Valid variable Names

count
_value
totalAmount
num_students
amazing_spiderman_2

Invalid variable Names

123abc (starts with digit)
my-variable (hyphen)
float (reserved keyword)

(b) **Types of constants in C**

The various types of constants in C programming are:

Integer constants: Represent integers & can be represented as decimal, octal or hexadecimal format or even binary.

Eg: 123, 0345, 0x123A, 0b1010

Floating point constants: Represent real numbers with fractional parts. They can be expressed in decimal or exponential notation.

Eg. 3.14, 4e-3

Character & string constants: Character constants are single characters in single quotes, whereas string characters are in double quotes terminated by null character '\0'.

Eg. 'a', 'b', '\n', "ram", "shyam".

Symbolic constants: are defined using #define preprocessor directive or const keyword their value doesn't change

Eg. #define PI 3.14, const int NUM = 100;

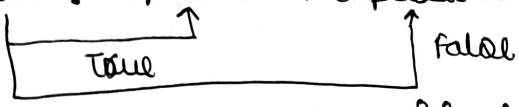
Enumeration constants: are named integers defined using enum keyword.

enum Weekday { Monday, Tuesday, Wednesday, Thursday, Friday };

Name : Deepankar Sharma
Course : MCA
Student ID : 233512013

Ques③ (a) Ternary Operator

condition ? expression1 : expression2



Ternary operator is also called conditional operator, where first operand is condition, if it is true expression1 is executed otherwise expression2 is executed. for example:

```
int x = 10;
int result = (x > 5) ? 100 : 200;
```

↑ 100 because expression 1 is evaluated.

a	b	AND	OR	XOR
1	0	0	1	1
0	1	0	1	1
0	0	0	0	0
1	1	1	1	0

(b)
```
#include <stdio.h>
int main() {
    // Logical AND, OR, & Bitwise XOR
    int x = 1, y = 0;

    printf("%d \n", 1 && 0);  // 0 because 1 AND 0 = 0
    printf("%d \n", 1 || 0);  // 1 because 1 OR 0 = 1

    printf("%d \n", 5 ^ 8);  // 13 because 5 ^ 8 = 13
                             (0101) (1000) (1101)

    return 0;
}
```

AND (logical) [&&] → It is used for representing the logical AND operation between two operands.

OR (logical) [||] → It is used for representing logical OR operation between two operands

Bitwise XOR (^) → Used to perform logical XOR, between each bits of two operands.

Name: Deepankar Sharma
Student ID: 23351201 3
Course: MCA

Ques④ (a) **while loop v/s do-while loop**

While loops are entry control loops, where a condition is checked before entering the loop whereas do-while are exit control loops where the condition is checked once the loop body is executed. do while loops can be used for menu driven programs.

while

```
int true = 10;
while (true) {
    true--;
}
```

if false the loop will never execute

```
int choice;
do {
    printf ("Enter choice ");
    scanf ("%d " & choice);
} while (choice);
```

The loop body will execute at least once.

(b) **C Program for break & continue**

```
#include <stdio.h>
int main() {
    for (int i=0; i<100; i++) {
        if (i%2==0) { continue; } skip even numbers
        if (i==9) { break; } // break once 9 is reached
        printf ("%d__", i);
    }
    return 0;
}
```

// Output : 1_3_5_7_

Break : Break statement is used to break a code execution block permanently. Therefore once i=9, the loop breaks.

Continue : continue doesn't break code block permanently. It breaks at code executed as for and skips to next iteration.

Name: Deepankar Sharma
Course: MCA
Student ID: 233512013

Ques ⑤ : User defined Function

(a)

User defined functions are parts of code that perform specific tasks and can be called from other parts of the program.

Prototype : declares return type, function name & parameters without providing actual implementation

```
void hello();
```

void → return type; hello → function name; () → no parameters

definition : actual implementation of the function

```
void hello() {
    printf("This is hello function");
}
```

call : calling a user defined function is same as any function, write name & pass comma separated parameters in parantheses.

```
hello(); // function call.
```

(b) Pointers

A pointer in C is a special variable that stores address of another variable. It allows direct access to memory location.

Advantages of pointers

① Direct access to memory location enables efficient manipulation of data & dynamic memory allocation.

② Pointers allow functions to pass variables through reference & allowing operations to execute on original variables without having to create their dummies.

③ Pointer arithmetic allows faster and efficient of array data types.

④ Pointers also provide mechanism for accessing hardware resources, memory mapped I/O operations, making them very suitable for system level programming.