Internal Assessment Test I – November 2025

Sub Code: 1BPLC205E Branch: ECE Sem/Sec: I/ M, N,O & P

SOLUTIONS

1a. You are asked to design a program that calculates the area of a rectangle using flowcharts. Create the flowchart and corresponding algorithm.

Algorithm to calculate the area of a rectangle:

Step-1 : **Start.**

Step2: **Declare variables**: for length, width, and area. These variables should be of a floating-point data type (e.g., float or double) to handle potential decimal values.

Step-3: **Prompt the user**: to enter the length of the rectangle.

Step-4: **Read the length**: value entered by the user and store it in the length variable.

Step-5: **Prompt the user**: to enter the width of the rectangle.

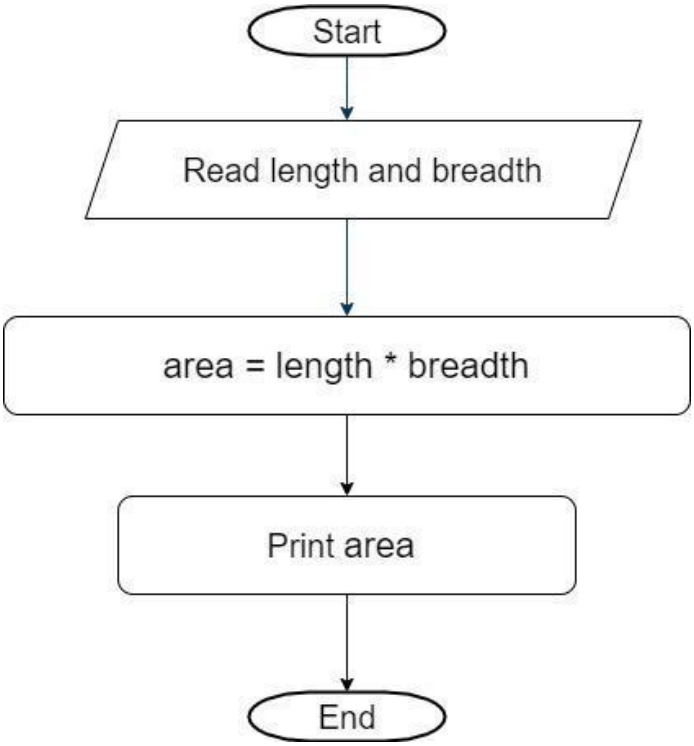
Step-6: **Read the width**: value entered by the user and store it in the width variable.

Step-7: **Calculate the area**: using the formula: area = length * width.

Step-8: **Display the calculated area**: to the user.

Step-9: **End.**

Flowchart:



Program

```
#include <stdio.h> // Required for input/output functions like printf and scanf
int main() {
    float length, width, area; // Declare float variables for length, width, and area
    // Prompt the user to enter the length
    printf("Enter the length of the rectangle: ");
    // Read the length entered by the user
    scanf("%f", &length);
    // Prompt the user to enter the width
    printf("Enter the width of the rectangle: ");
    // Read the width entered by the user
    scanf("%f", &width);
    // Calculate the area
    area = length * width;
    // Display the calculated area
    printf("The area of the rectangle is: %.2f\n", area); // .2f formats to two decimal places
    return 0; // Indicate successful program execution
```

}

Output:

Enter the length of rectangle: 4.2

Enter the width of the rectangle: 5.3

The area of the rectangle is 22.26

1.b. Write a C program, a company pays bonus only if an employee has completed more than 5 years of service

```
#include <stdio.h>
int main() {
    // Declare variables for salary, service years, and bonus amount
    float salary, bonus;
    int years_of_service;
    // Prompt the user to enter the annual salary
    printf("Enter employee's annual salary: ");
    scanf("%f", &salary);
    // Prompt the user to enter the years of service
    printf("Enter years of service: ");
    scanf("%d", &years_of_service);
    // Check if the employee has completed more than 5 years of service
    if (years_of_service > 5) {
        // Calculate the bonus as 5% of the salary
        bonus = salary * 0.05;
```

```
        // Display the bonus amount
        printf("Bonus amount: %.2f\n", bonus);
    } else {
        // If the condition is not met, print a message indicating no bonus
        printf("No bonus is applicable for this employee.\n");
    }
    return 0;
}
```

Output:

```
Enter employee's annual salary: 30000
Enter years of service: 7
Bonus amount: 1500
Enter employee's annual salary: 15000
Enter years of service: 3
No bonus is applicable for this employee.
```

1. a.Write an algorithm and draw a flowchart for calculating the factorial of a given number.

   Algorithm for Calculating Factorial

   Step-1: Start.

   Step-2: Read: the input number, n.

   Step-3: Initialize: variables: factorial = 1 and i = 1.

   Step-4: Check: if n is less than 0. If true, display an error message ("Factorial is not defined for negative numbers") and stop.

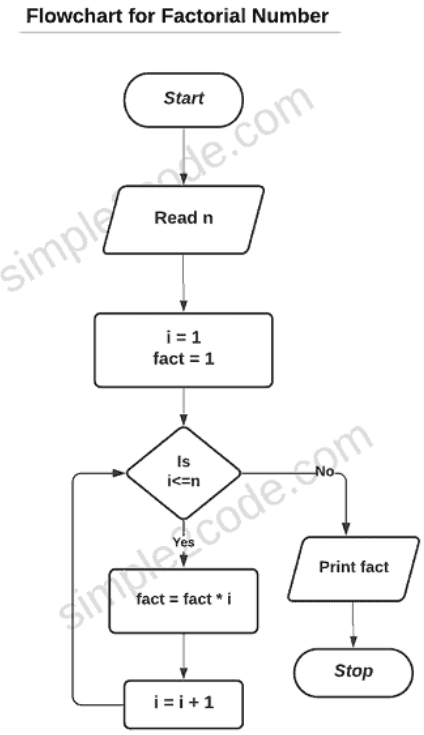
   Step-5: Loop: while i is less than or equal to n:

   a. Calculate factorial = factorial * i.
   b. Increment i by 1 (i = i + 1).

   Step-6: Print: the value of factorial.

   Step-7: Stop.

   Flowchart



2. b. Write notes on Constants (Integer and Real Numbers)
   Constants in C represent fixed values that a program cannot alter during its execution. These are also known as literals.
   Integer Constants
   Integer constants are whole numbers without any fractional or decimal part.

- **Rules for Construction:**
  - Must have at least one digit.
  - Must not contain a decimal point.
  - Can be positive or negative; if no sign is present, it is assumed to be positive.
  - No commas or blank spaces are allowed within the constant.
- **Types:**
  - **Decimal:** Numbers without any prefix (e.g., 123, -45).
  - **Octal:** Numbers prefixed with 0 (e.g., 012 represents decimal 10).
  - **Hexadecimal:** Numbers prefixed with 0x or 0X (e.g., 0xFF represents decimal 255).
- **Suffixes:**
  - u or U: Denotes an unsigned integer (e.g., 100U).
  - l or L: Denotes a long integer (e.g., 123456L).
  - ll or LL: Denotes a long long integer (e.g., 987654321LL).

Real (Floating-Point) Constants
Real constants represent numbers with fractional parts, also known as floating-point numbers.

- **Rules for Construction:**
  - Must have at least one digit.
  - Must contain a decimal point.
  - Can be positive or negative; if no sign is present, it is assumed to be positive.
  - No commas or blank spaces are allowed within the constant.
- **Forms:**
  - **Fractional Form:** A sequence of digits, a decimal point, and another sequence of digits (e.g., 3.14, -0.5, .75).
  - **Exponential (Scientific) Form:** Used for very large or very small numbers. It consists of a mantissa, followed by e or E, and an exponent (e.g., 2.5e-3 for 0.0025, 1.23E5 for 123000). The mantissa can be an integer or a fractional form, and the exponent must be an integer.
- **Default Type:**
  Real constants are typically treated as double by default. To explicitly specify a float constant, append f or F (e.g., 3.14f). To specify a long double, append l or L (e.g., 1.234L).

3. a. Write a C code using #define preprocessor directive

```
#include <stdio.h>

// Define a symbolic constant for the maximum number of items

#define MAX_ITEMS 10

// Define a function-like macro to calculate the square of a number

#define SQUARE(x) ((x) * (x))

int main() {

   int item_count = 5;

   float value = 3.5;

   // Using the symbolic constant

   printf("Maximum allowed items: %d\n", MAX_ITEMS);

   // Using the symbolic constant in a conditional statement

   if (item_count < MAX_ITEMS) {

      printf("You can add %d more items.\n", MAX_ITEMS - item_count);

   } else {

      printf("Maximum items reached.\n");

   }

   // Using the function-like macro
```

```
    printf("The square of %.2f is %.2f\n", value, (float)SQUARE(value));
    return 0;
}
```

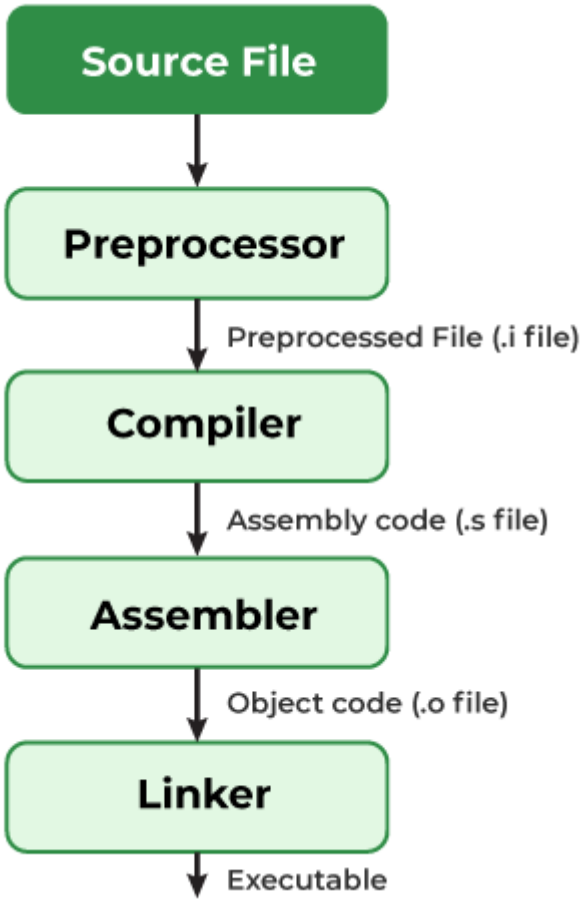
Output

```
Maximum allowed items: 10
You can add 5 more items.
The square of 3.50 is 12.25
```

3.b.With a neat diagram, explain the steps in the compilation and execution of a C program.

The compilation is the process of converting the source code of the C language into machine code. As C is a mid-level language, it needs a compiler to convert it into an executable code so that the program can be run on our machine.

The C program goes through the following phases during compilation:



Compilation Process in C

Understanding the compilation process in C helps developers optimize their programs.

How do we compile and run a C program?

We first need a compiler and a code editor to compile and run a C Program. The below example is of an Ubuntu machine with GCC compiler.

Step 1: Creating a C Source File

We first create a C program using an editor and save the file as filename.c In linux, we can use vi to create a file from the terminal using the command:

vi filename.c

In windows, we can use the Notepad to do the same. Then write a simple hello world program and save it.

Step 2: Compiling using GCC compiler

We use the following command in the terminal for compiling our **filename.c** source file.

gcc filename.c –o filename

We can pass many instructions to the GCC compiler to different tasks such as:

- The option -Wall enables all compiler's warning messages. This option is recommended to generate better code.
- The option -o is used to specify the output file name. If we do not use this option, then an output file with the name a.out is generated.

If there are no errors in our C program, the executable file of the C program will be generated.

Step 3: Executing the program

After compilation executable is generated and we run the generated executable using the below command.

./filename *// for linux*

filename *// for windows*

4.a.Develop a C program to find the largest of three numbers using ternary operator.

```
#include <stdio.h>
int main() {
    int num1, num2, num3, largest;
    // Prompt the user to enter three numbers
    printf("Enter three numbers: ");
    scanf("%d %d %d", &num1, &num2, &num3);
    // Use the ternary operator to find the largest number
```

```
    // First, compare num1 and num2. The larger of these two is then compared with num3.
    largest = (num1 > num2) ? ((num1 > num3) ? num1 : num3) : ((num2 > num3) ? num2 : num3);
    // Print the largest number
    printf("The largest number among %d, %d, and %d is: %d\n", num1, num2, num3, largest);
    return 0;
}
```

Output

Enter three integers: 5 12 8

The largest number is: 12

4.b. List all the Operators in C and evaluate the following expression: i) x = a + b / 5 – c * 4 – 1 where a = 6, b = 12, c = 5 b.ii) 10 ! = 10 || 5 < 4 && 8.

i) x = a + b / 5 – c * 4 – 1

where a = 6, b = 12, c = 5

x=6+12/5-5*4-1

=6+2.4-5*4-1

=6+2.4-20-1

=8.4-20-1

=-12.6

ii) 10!=10||5<4&&8

Step-by-step:

1. Evaluate 10 != 10
   10 != 10 → **false (0)**
2. Evaluate 5 < 4
   5 < 4 → **false (0)**
3. Now the right side is (0 && 8)
   Evaluate 0 && 8:
   - Left operand is 0 (false), so && short-circuits: result is **0** and the 8 is **not evaluated** (though 8 is a nonzero value that would be true if evaluated).
   - 0 && 8 → **0**
4. Now overall: 0 || 0 → **0**
5. Final result: **0** (false)

5.a. Explain with syntax, if and if-else statements in C program.

Decision making may have to change the order of execution of statements based on certain conditions, or repeat a group of statements until certain specified conditions are met

C language possesses such decision-making capabilities by supporting the following statement

1. if statement
2. Switch statement
3. Conditional operator statement
4. Goto statement

These statements are popularly known as decision-making statements. Since these statements 'control' the flow of execution, they are also known as control statements

The if statement is a powerful decision-making statement and is used to control the flow of execution of statement.

It is basically two way decision statement and is used in conjunction with an expression

if(test expression)

Syntax

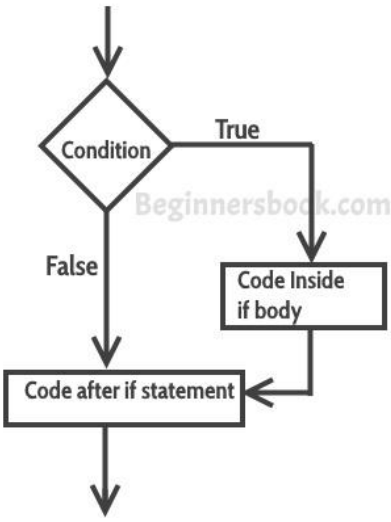
```
if(Condition)
{
Statement block;
}
Statement x;
```

If the condition is true, the statement block will be executed; otherwise the statement block is skipped and execution will jump to the statement-x

When the condition is true, both statement block will be executed.

Example:

```
if(category==sports)
{
marks= marks+bonumark;
}
printf("%f",marks);
```



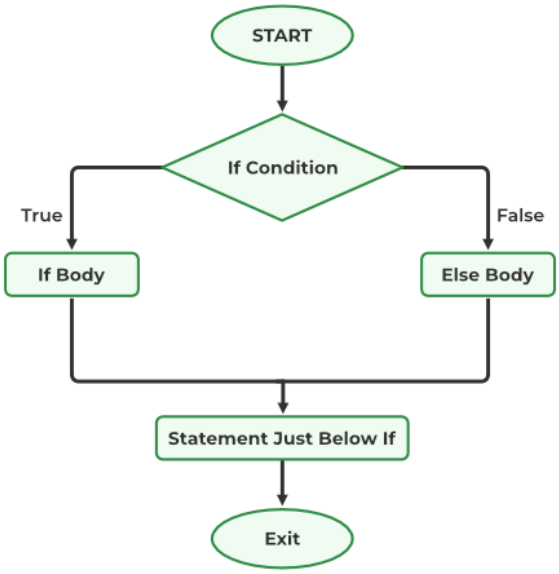

```
#include <stdio.h>
int main() {
   int age = 20;
   if (age >= 18) {
      printf("You are eligible to vote.\n");
   }
   return 0;
}
```

If-else

The if-else is an extension of the simple if statement

Syntax

```
if(cond)
{
True block statement
}
else
{
False statement
}
Statement-x;
```




```
if(code==1)
boy=boy+1;
else
girl=girl+1;
```

```
#include <stdio.h>

int main() {
    int number = 7;
    if (number % 2 == 0) {
        printf("%d is an even number.\n", number);
    } else {
        printf("%d is an odd number.\n", number);
    }
    return 0;
}
```

5.b.Geometric Shape Area Calculator : Menu to calculate area of:

➢ Circle

➢ Rectangle

➢ Triangle

➢ Square

```
#include <stdio.h>
#include <math.h>   // for M_PI (value of π)
int main() {
    int choice;
    float radius, length, breadth, base, height, side, area;
    // Display menu
    printf("===== GEOMETRIC SHAPE AREA CALCULATOR =====\n");
    printf("1. Circle\n");
    printf("2. Rectangle\n");
    printf("3. Triangle\n");
    printf("4. Square\n");
    printf("Enter your choice (1-4): ");
    scanf("%d", &choice);
    switch (choice) {
        case 1: // Circle
            printf("Enter radius of circle: ");
            scanf("%f", &radius);
            area = M_PI * radius * radius;
            printf("Area of Circle = %.2f\n", area);
            break;
        case 2: // Rectangle
            printf("Enter length and breadth of rectangle: ");
            scanf("%f %f", &length, &breadth);
            area = length * breadth;
            printf("Area of Rectangle = %.2f\n", area);
```

```
            break;
        case 3: // Triangle
            printf("Enter base and height of triangle: ");
            scanf("%f %f", &base, &height);
            area = 0.5 * base * height;
            printf("Area of Triangle = %.2f\n", area);
            break;
        case 4: // Square
            printf("Enter side of square: ");
            scanf("%f", &side);
            area = side * side;
            printf("Area of Square = %.2f\n", area);
            break;
        default:
            printf("Invalid choice! Please select between 1 and 4.\n");
    }
    return 0;
}
```

6.a. Write a C Program to find the Sum on N Numbers using for-loop.

```
#include <stdio.h>
int main() {
    int n, i, num;
    int sum = 0;
    // Step 1: Get how many numbers to add
    printf("Enter how many numbers you want to add: ");
    scanf("%d", &n);
    // Step 2: Use for loop to read and add numbers
```

```
    for (i = 1; i <= n; i++) {
        printf("Enter number %d: ", i);
        scanf("%d", &num);
        sum = sum + num;   // accumulate the sum
    }
    // Step 3: Display the total sum
    printf("The sum of %d numbers is: %d\n", n, sum);
    return 0;
}
```

Output:

Enter how many numbers you want to add: 4

Enter number 1: 10

Enter number 2: 20

Enter number 3: 30

Enter number 4: 40

The sum of 4 numbers is: 100

6.b. State the differences between the syntax of while and do-while loops and explain how it affects program execution

1. while

```
while (condition)
{
    // loop body
    // statements to be executed repeatedly
}
```

The **condition is checked first**.

If it's **true**, the loop body executes.

After execution, the condition is checked again.

The loop stops when the condition becomes **false**.

The loop **may not execute even once** if the condition is **false initially**.

Syntax of do-while Loop

```
do
{
   // loop body
   // statements to be executed repeatedly
} while (condition);
```

The **loop body executes first**, **then** the condition is checked.

If the condition is **true**, it repeats the loop.

If **false**, the loop stops.

The loop **executes at least once**, even if the condition is **false initially**.

7. An elevator goes from floor 1 to floor. Write a program using a loop to print: Floor 1 reached Floor 2 reached Floor 3 reached Floor 4 reached.

```
#include <stdio.h>
int main() {
   int floor;
   // Loop from floor 1 to 4
   for (floor = 1; floor <= 4; floor++) {
      printf("Floor %d reached\n", floor);
   }
   return 0;
}
```