Bangladesh University of Business & Technology (BUBT)

Lab Report

Course Title	: Structured Programming Language Lab
Course Code	: CSE 102

Submitted by

Name	: Makhzum-Bin-Harun
ID	: 20254103279
Intake	: 56
Section	: 07
Program:	B.Sc. Engg. in CSE

Submitted to

Name : Sourav Kundu
Designation: Lecturer
Department of Computer Science & Engineering
Bangladesh University of Business & Technology.

Date of Submission: September 03, 2025

Signature of Teacher

Table of Contents

Problem 01. Write a C program to show the below text:

My name is Mr. Doe,
I am from USA,
My employee id : 4263

01.1. Algorithm: Algorithm to Print three lines given in the following problem

1. Start
2. Include the header file **`<stdio.h>`**
3. Define the **`main()`** function
4. Use **`printf()`** to display:

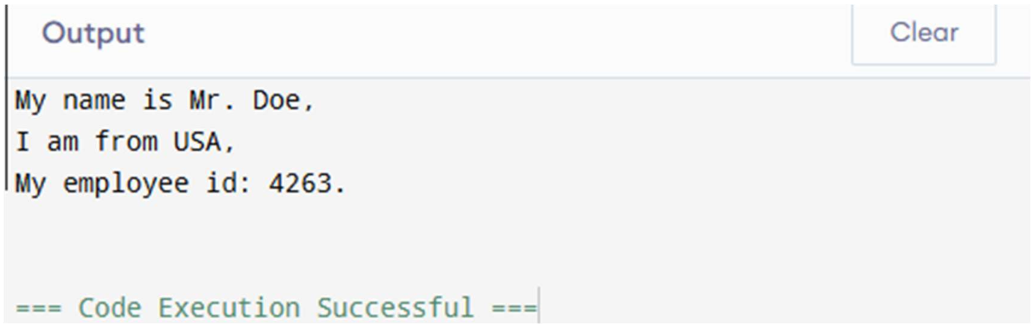
```
My name is Mr. Doe,

I am from USA,

My employee id: 4263.
```

5. End the program with **`return 0;`**
6. Stop

01.2. C Program Code:

```c
#include <stdio.h>

int main() {
    printf("My name is Mr. Doe,\n");
    printf("I am from USA,\n");
    printf("My employee id: 4263.\n");
    return 0;

}
```

01.3. Output:



```
My name is Mr. Doe,
I am from USA,
My employee id: 4263.


=== Code Execution Successful ===
```

01.4. Discussion:

In this program, we display text on the screen using the `printf()` function in C. The header file `<stdio.h>` is required since it contains the definition of input-output functions. The program execution begins with the `main()` function. Each line of text is printed with the help of `printf()`, and the escape sequence `\n` is used to move the cursor to the next line so that each statement appears in a separate line. Finally, `return 0;` indicates that the program has executed successfully.

Problem 02. Suppose you have 500 Tk, you have to buy a product cost of 350.80 Tk, and later you got 200 Tk from your brother.
Write a C program where user can input the balance and finally can see how much money remain?

02.1. Algorithm: Algorithm to find remaining balance

1. Start
2. Declare three variables: `balance`, `product_cost`, and `money_received` (type float).
3. Ask the user to input the initial balance.
4. Ask the user to input the product cost.
5. Ask the user to input the amount received from the brother.
6. Calculate the remaining balance using the formula:

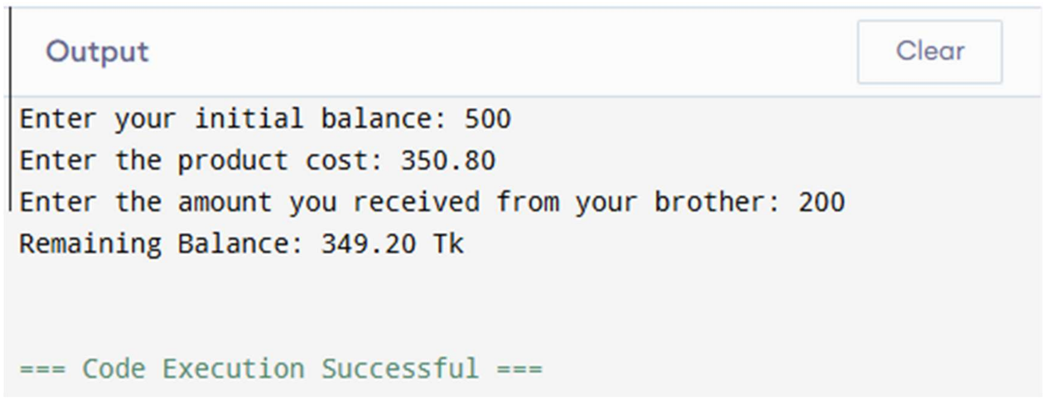
   ```
   balance = balance - product_cost + money_received
   ```

7. Display the remaining balance (with 2 decimal places).
8. Stop

02.2. C Program Code:

```c
#include <stdio.h>

int main() {
    float balance, product_cost, money_received;
    printf("Enter your initial balance: ");
    scanf("%f", &balance);
    printf("Enter the product cost: ");
    scanf("%f", &product_cost);
    printf("Enter the amount you received from your brother: ");
    scanf("%f", &money_received);
    balance = (balance-product_cost+money_received);
    printf("Remaining Balance: %.2f Tk\n", balance);

    return 0;
}
```

02. 3. Output:



```
Enter your initial balance: 500
Enter the product cost: 350.80
Enter the amount you received from your brother: 200
Remaining Balance: 349.20 Tk


=== Code Execution Successful ===
```

02.4. Discussion:

In this program, we calculate the remaining balance after a purchase and receiving extra money. We first declare three floating-point variables to store the user's initial balance, product cost, and money received. Floating-point (float) is used because the values may include decimals (e.g., 350.80).

The program uses `scanf()` to take user inputs and `printf()` to display results. The calculation is performed using the formula:

```
Remaining Balance = Initial Balance - Product Cost + Money Received
```

Finally, the result is printed using `%.2f`, which ensures that the balance is shown up to two decimal places.

For example, if the inputs are:

```
Enter your initial balance: 500

Enter the product cost: 350.80

Enter the amount you received from your brother: 200
```

The calculation becomes:

```
Remaining Balance: 349.20 Tk
```

Thus, the program works correctly for different inputs and gives the remaining balance.

Problem 03. Write a C program to find if a number is positive or not?
Take number as input from keyboard and show output as below:

    if positive:
        5 is a positive number.

    If not:
        -5 is not a positive number.

03.1. Algorithm: Algorithm to Convert Days into Years, Weeks, and Days

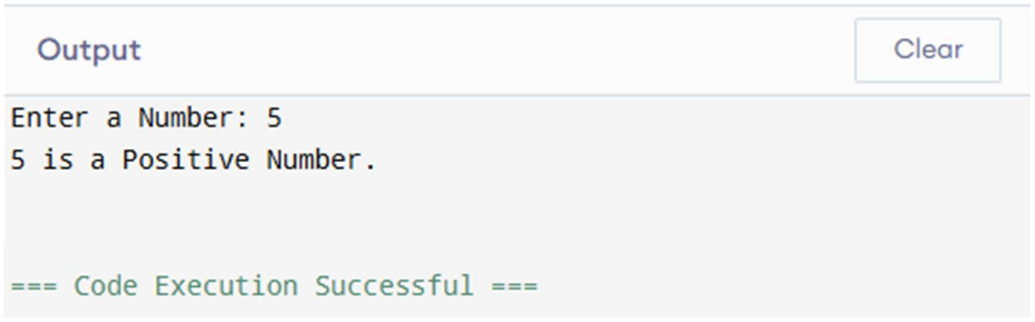
1. Start
2. Declare an integer variable `num`.
3. Prompt the user to enter a number.
4. Take the number as input using `scanf()`.
5. Check the condition:
   - If `num > 0`, print "`num is a Positive Number.`"
   - Else if `num == 0`, print "`num is neither Positive nor Negative.`"
   - Else (when `num < 0`), print "`num is not a Positive Number.`"
6. Stop

03.2. C Program Code:

```
#include <stdio.h>

int main() {
    int num;
    printf("Enter a Number: ");
    scanf("%d", &num);

    if (num > 0) {
        printf("%d is a Positive Number.\n", num);
    }
    else if(num==0) {
        printf("%d is neither Positive nor Negative.\n", num);
    }
    else {
        printf("%d is not a Positive Number.\n", num);
    }

    return 0;
}
```

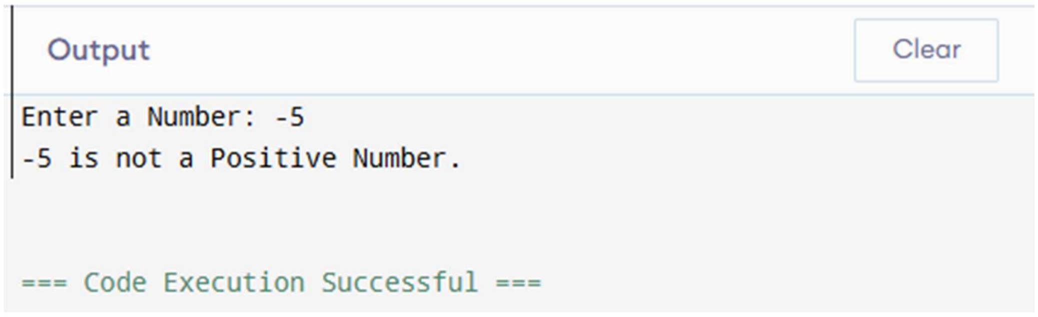
03.3. Output:

Case 01:



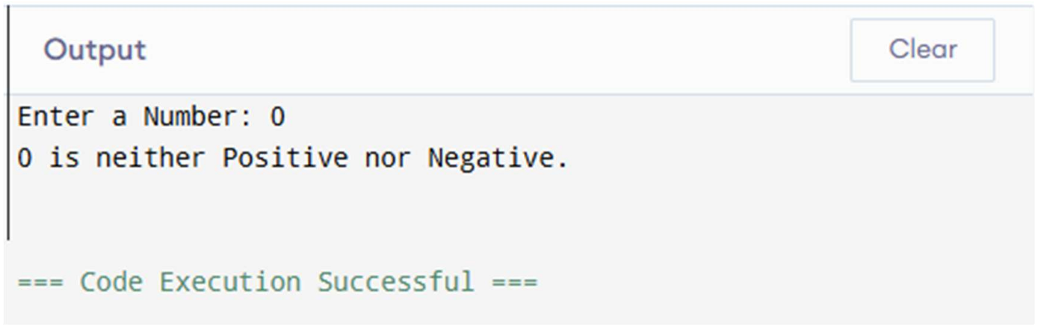
```
Enter a Number: 5
5 is a Positive Number.


=== Code Execution Successful ===
```

Case 02:



```
Enter a Number: -5
-5 is not a Positive Number.


=== Code Execution Successful ===
```

Case 03:



```
Enter a Number: 0
0 is neither Positive nor Negative.


=== Code Execution Successful ===
```

03.4. Discussion:

In this program, we check whether a given number is positive or not. The program starts by declaring an integer variable `num` to store the input value. Using `scanf()`, the user provides a number.

- If the number is greater than 0, it is considered positive, and the program prints:

  ```
  num is a Positive Number.
  ```

- If the number is exactly 0, it is neither positive nor negative, so the program prints:

  ```
  0 is neither Positive nor Negative.
  ```

- If the number is less than 0, it is not positive, and the program prints:

  ```
  num is not a Positive Number.
  ```

The use of if-else-if conditions ensures that only one correct output is printed depending on the input.

Problem 04. Write a C program to input two numbers from user and find power without using `pow()` function.

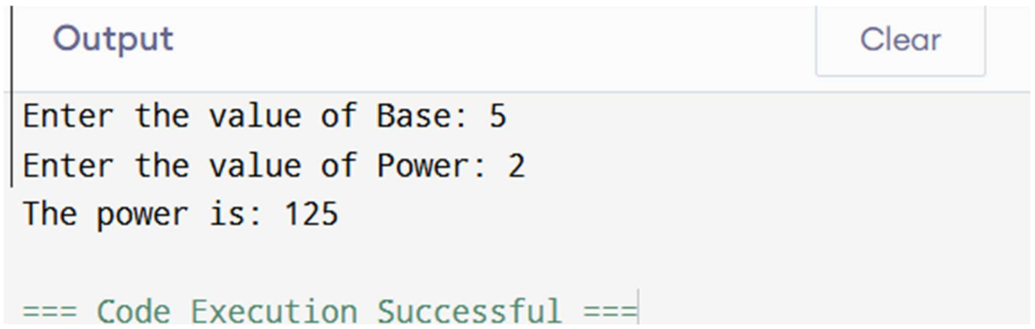
04.1. Algorithm: Algorithm to Find the power without using `pow()` function

```
1. Start
2. Declare integer variables: B (Base), P (Power), POWER (Result).
3. Initialize POWER=1.
4. Ask the user to input the values of B and P.
5. Use a for loop to multiply the base repeatedly:
    • Repeat from i=1 to P.
    • Multiply POWER = POWER * B.
6. After the loop ends, POWER will store the result of $B^P$.
7. Print the final result.
8. Stop
```

04.2. C Program Code:

```
#include<stdio.h>

int main() {
     int B, P, N, POWER=1;
     printf("Enter the value of Base: ");
     scanf("%d",&B);
     printf("Enter the value of Power: ");
     scanf("%d",&P);

     for(int i=1; i<=P; i++) {
     POWER*=B;
     }

     printf("The power is: %d", POWER);

     return 0;

}
```

04.3. Output:



```
Enter the value of Base: 5
Enter the value of Power: 2
The power is: 125

=== Code Execution Successful ===
```

04.4. Discussion:

In this program, we compute the power of a number using a **loop** instead of the library function `pow()`.

- We first declare the necessary variables:
    - B for base,
    - P for power,
    - `POWER` to store the result (initialized to 1).
- The program then asks the user to input both base and power.
- Using a **for loop**, the base B is multiplied repeatedly P times.
    - Example: If `B = 2` and `P = 3`, the calculation becomes:

$$\text{POWER} = 1 \times 2 \times 2 \times 2 = 8$$

- The final result is printed using `printf()`.

Problem 05. Write a C program to find the sum of all odd numbers from 1 to n using a do-while loop.

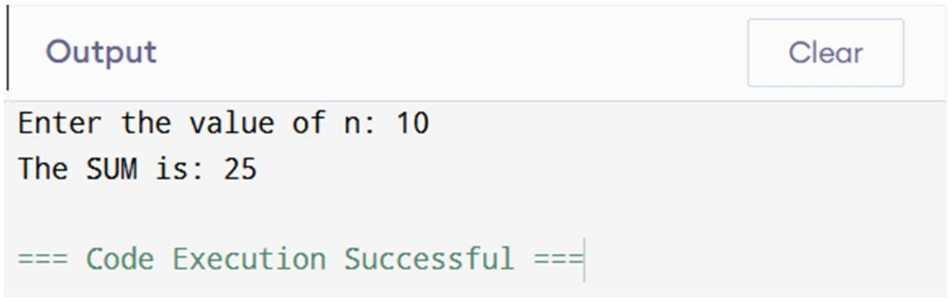
05.1. Algorithm: Algorithm to Convert Days into Years, Weeks, and Days

1. Start
2. Declare integer variables: `n`, `i`, and `SUM`.
3. Initialize `i=1` and `SUM=0`.
4. Take input for `n` (the upper limit).
5. Use a **do-while loop**:
    - Check if `i` is odd (`i%2!=0`).
    - If odd, add `i` to `SUM`.
    - Increment `i`.
6. Repeat the loop until `i<= n`.
7. Print the value of `SUM`.
8. Stop

05.2. C Program Code:

```
#include <stdio.h>

int main() {
    int n, i = 1, SUM = 0;
    printf("Enter the value of n: ");
    scanf("%d", &n);

    do {
        if (i % 2 != 0) {
            SUM += i;
        }
        i++;
    } while (i <= n);

    printf("The SUM is: %d", SUM);
    return 0;
}
```

05.3. Output:



```
Enter the value of n: 10
The SUM is: 25

=== Code Execution Successful ===
```

05.4. Discussion:

In this program, we calculate the sum of all **odd numbers** between 1 and `n` using a **do-while loop**.

- First, the program initializes `i = 1` and `SUM = 0`.
- The user enters a positive integer `n`.
- The **do-while loop** executes at least once and keeps running until `i <= n`.
- Inside the loop:
    - If the number `i` is odd (`i%2!=0`), it is added to SUM.
    - The variable `i` is then incremented.
- Finally, the program prints the sum of all odd numbers from 1 to `n`.

Problem 06. Write a C program to input a number (5) and calculate its factorial using for loop.

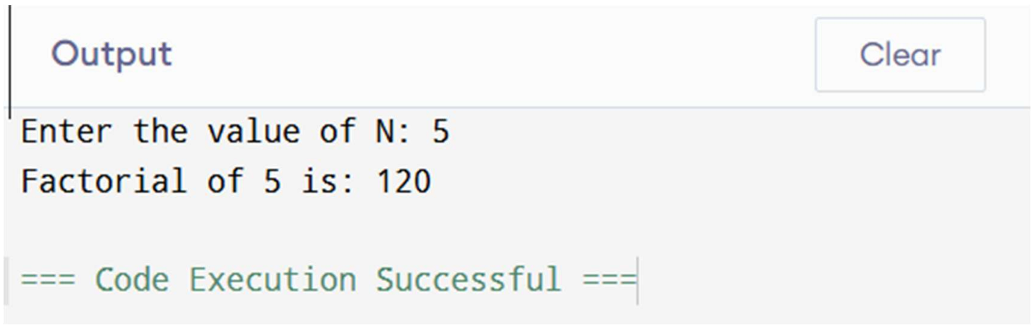
06.1. Algorithm: Algorithm to Factorial using For Loop

```
1. Start
2. Declare integer variables N and FACTORIAL (initialize FACTORIAL=1).
3. Display a message: "Enter the value of N:".
4. Take input value of N from user.
5. Repeat a loop from i = 1 to N:
     • Multiply FACTORIAL by i.
6. After loop ends, display the result: "Factorial of N is: FACTORIAL".
7. End
```

06.2. C Program Code:

```
#include<stdio.h>

     int main() {
     int N, FACTORIAL=1;

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

     for(int i=1; i<=N; i++) {
     FACTORIAL*=i;
     }
     printf("Factorial of %d is: %d", N, FACTORIAL);

     return 0;

}
```

06.3. Output:



```
Enter the value of N: 5
Factorial of 5 is: 120

=== Code Execution Successful ===
```

06.4. Discussion:

The factorial of a number N (denoted as $N!$) is the product of all positive integers from 1 up to N.
For example:

$$5! = 5 \times 4 \times 3 \times 2 \times 1 = 120$$

In this program, the user enters a number `N`. The program initializes a variable `FACTORIAL` to 1 and then multiplies it by each integer from 1 to N using a for loop. This iterative process continues until the loop reaches N. Finally, the program displays the computed factorial.

The use of a loop makes the calculation efficient, avoiding repeated manual multiplications. This approach also works for any positive integer input by the user. However, factorial values grow very fast, so for large N (e.g., above 20), the result may exceed the storage capacity of an `int` data type. In such cases, a `long long` or other big number handling methods should be used.