

## PPS7

### Q1

#### **Aim:**

Write a function in 'C' to perform factorial of natural number 'n'. Get the user input for 'n'.

#### **Procedure:**

##### **Input:**

A natural number, n

##### **Output:**

Factorial of n

##### **Algorithm:**

Step 1: Declare 'factorial' function with return type 'int' and argument 'int n'

#### **Main Function**

Step 1: Declare variables n and r

Step 2: Read a natural number 'n' from user

Step 3: Call 'factorial' function and save its return value in r

Step 4: Display 'r' which is the factorial of 'n'

#### **Factorial Function (Recursion)**

Step 1: If n is equal to 0 or n is equal to 1  
return 1

Step 2: Else call the function again in return statement  
return n\*factorial(n-1)

## Code:

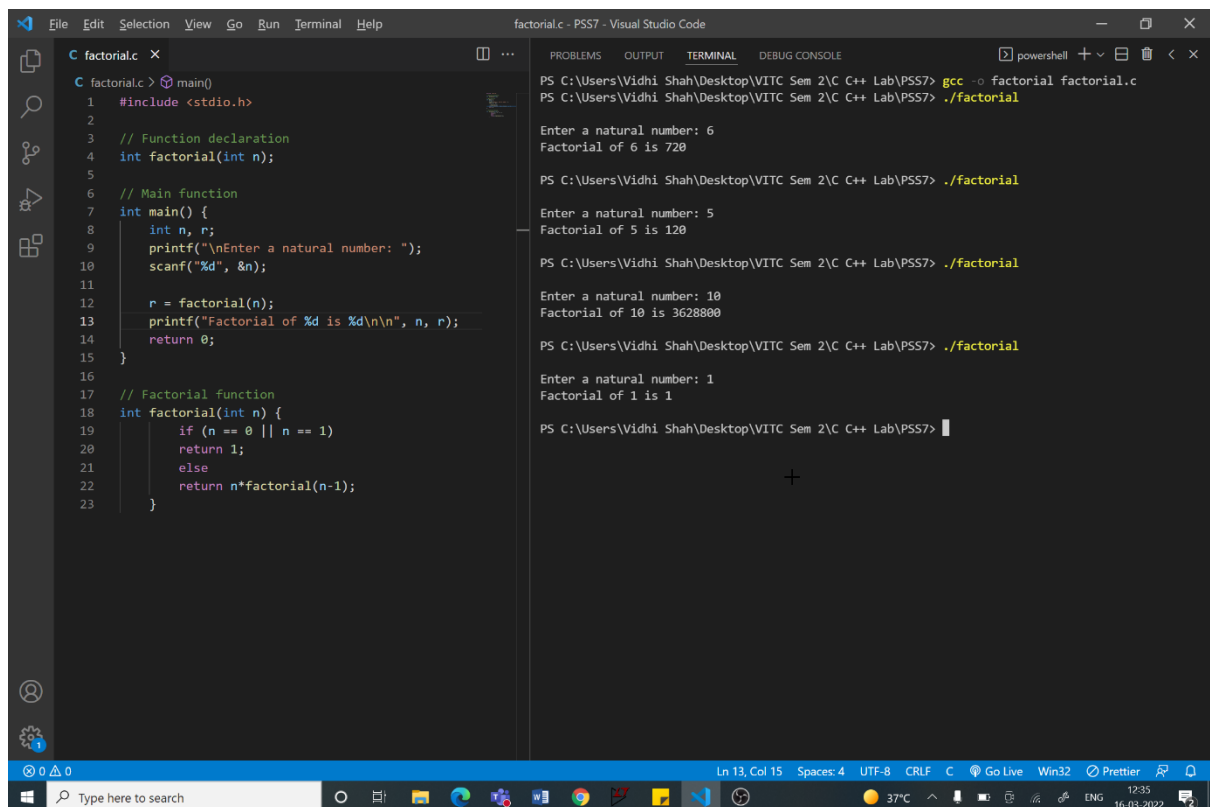
```
#include <stdio.h>

// Function declaration
int factorial(int n);

// Main function
int main() {
    int n, r;
    printf("\nEnter a natural number: ");
    scanf("%d", &n);

    r = factorial(n);
    printf("Factorial of %d is %d\n\n", n, r);
    return 0;
}

// Factorial function
int factorial(int n) {
    if (n == 0 || n == 1)
        return 1;
    else
        return n*factorial(n-1);
}
```



The screenshot displays the Visual Studio Code interface with the file 'factorial.c' open. The code is a C program that calculates the factorial of a natural number. The terminal window on the right shows the execution of the program using the 'gcc' compiler and the './factorial' command. The output shows the program correctly calculating the factorial for inputs 6, 5, 10, and 1.

```
factorial.c - PSS7 - Visual Studio Code

C factorial.c x
1 #include <stdio.h>
2
3 // Function declaration
4 int factorial(int n);
5
6 // Main function
7 int main() {
8     int n, r;
9     printf("\nEnter a natural number: ");
10    scanf("%d", &n);
11
12    r = factorial(n);
13    printf("Factorial of %d is %d\n\n", n, r);
14    return 0;
15 }
16
17 // Factorial function
18 int factorial(int n) {
19     if (n == 0 || n == 1)
20         return 1;
21     else
22         return n*factorial(n-1);
23 }

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
PS C:\Users\Vidhi Shah\Desktop\VITC Sem 2\C++ Lab\PSS7> gcc -o factorial factorial.c
PS C:\Users\Vidhi Shah\Desktop\VITC Sem 2\C++ Lab\PSS7> ./factorial

Enter a natural number: 6
Factorial of 6 is 720

PS C:\Users\Vidhi Shah\Desktop\VITC Sem 2\C++ Lab\PSS7> ./factorial

Enter a natural number: 5
Factorial of 5 is 120

PS C:\Users\Vidhi Shah\Desktop\VITC Sem 2\C++ Lab\PSS7> ./factorial

Enter a natural number: 10
Factorial of 10 is 3628800

PS C:\Users\Vidhi Shah\Desktop\VITC Sem 2\C++ Lab\PSS7> ./factorial

Enter a natural number: 1
Factorial of 1 is 1

PS C:\Users\Vidhi Shah\Desktop\VITC Sem 2\C++ Lab\PSS7> |
```

## Q2

### **Aim:**

Write a function using 'C' to find the square of a number given by the user.

### **Procedure:**

#### **Input:**

An integer number, n

#### **Output:**

Square of number n

#### **Algorithm:**

Step 1: Declare 'square' function with return type 'int' and argument 'int n'

### **Main Function**

Step 1: Declare variables n and r

Step 2: Read an integer number 'n' from user

Step 3: Call 'square' function and save its return value in r

Step 4: Display 'r' which is the square of 'n'

### **Square Function**

Step 1: Return  $n*n$

## Code:

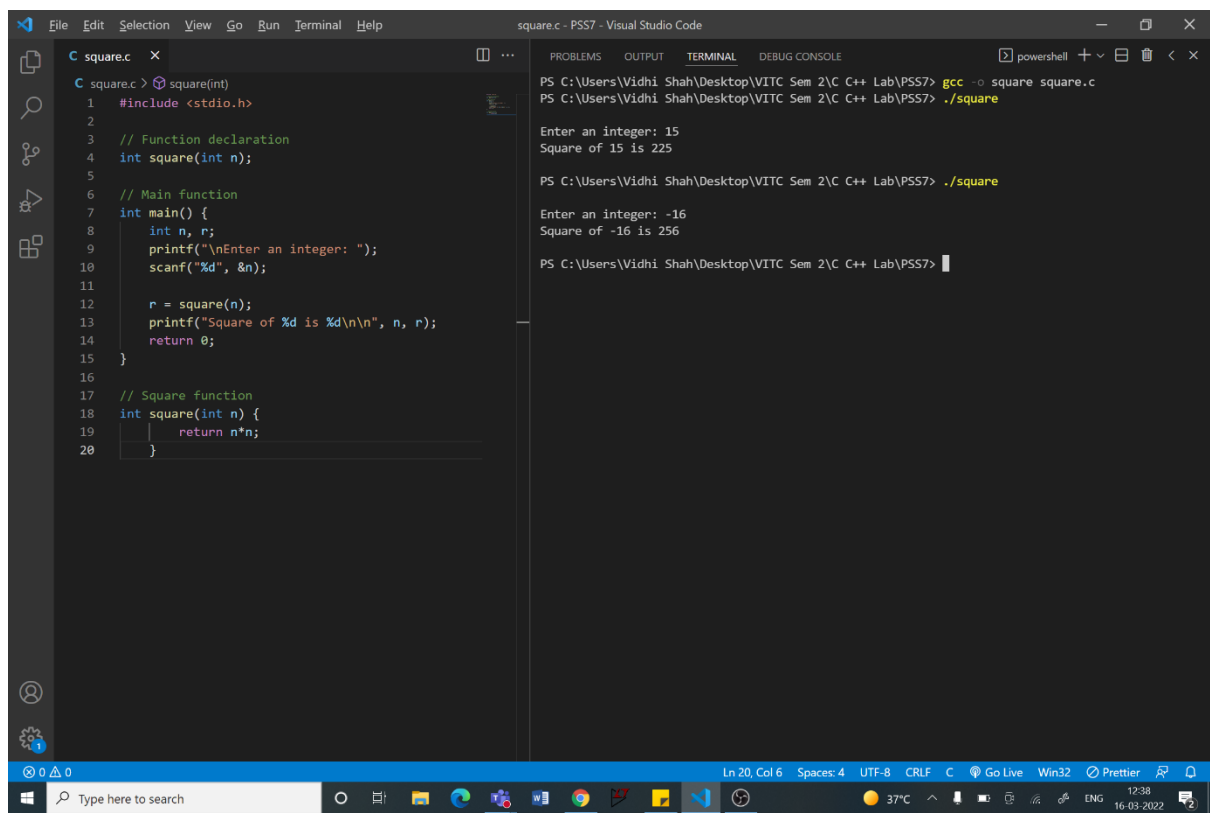
```
#include <stdio.h>

// Function declaration
int square(int n);

// Main function
int main() {
    int n, r;
    printf("\nEnter an integer: ");
    scanf("%d", &n);

    r = square(n);
    printf("Square of %d is %d\n\n", n, r);
    return 0;
}

// Square function
int square(int n) {
    return n*n;
}
```



The screenshot displays the Visual Studio Code editor with a C program named 'square.c' open. The code is identical to the one shown in the previous block. The terminal window on the right shows the execution of the program using the command 'gcc -o square square.c' followed by './square'. The program prompts the user to 'Enter an integer: 15' and outputs 'Square of 15 is 225'. A second run shows the user entering '-16' and the program outputting 'Square of -16 is 256'. The status bar at the bottom indicates the file is 'Ln 20, Col 6' with 4 spaces, using UTF-8 encoding and CRLF line endings. The system tray at the very bottom shows the date as 16-03-2022 and the time as 12:38.

```
square.c - PSS7 - Visual Studio Code

C square.c x
C square.c > square(int)
1 #include <stdio.h>
2
3 // Function declaration
4 int square(int n);
5
6 // Main function
7 int main() {
8     int n, r;
9     printf("\nEnter an integer: ");
10    scanf("%d", &n);
11
12    r = square(n);
13    printf("Square of %d is %d\n\n", n, r);
14    return 0;
15 }
16
17 // Square function
18 int square(int n) {
19     return n*n;
20 }
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

PS C:\Users\Vidhi Shah\Desktop\VITC Sem 2\C++ Lab\PSS7> gcc -o square square.c  
PS C:\Users\Vidhi Shah\Desktop\VITC Sem 2\C++ Lab\PSS7> ./square

Enter an integer: 15  
Square of 15 is 225

PS C:\Users\Vidhi Shah\Desktop\VITC Sem 2\C++ Lab\PSS7> ./square

Enter an integer: -16  
Square of -16 is 256

PS C:\Users\Vidhi Shah\Desktop\VITC Sem 2\C++ Lab\PSS7>

Ln 20, Col 6 Spaces: 4 UTF-8 CRLF C Go Live Win32 Prettier

Type here to search 37°C 12:38 16-03-2022

### Q3

#### **Aim:**

Write a 'C' Program to perform Matrix Addition for n x n Matrix. Get the user input for 'n'.

#### **Procedure:**

##### **Input:**

A natural number, n

Matrix 1 and Matrix 2 of dimension n x n

##### **Output:**

Matrix addition of matrix 1 and matrix 2

##### **Algorithm:**

Step 1: Declare variables n, i, j

Step 2: Declare 'matrixaddn' function with return type 'void' and argument of 2 integer matrices

##### **Main Function**

Step 1: Read a natural number 'n' from user

Step 2: Declare 2 integer matrices, 'm1' and 'm2', of dimension n x n

Step 2: Read and store elements of matrix 1, m1, using for loop

Step 3: Read and store elements of matrix 2, m2, using for loop

Step 4: Call 'matrixaddn' function

##### **Matrix Addition Function**

Step 1: Declare matrix 'ma' of dimension n x n

Step 2: Initialise i to 0. For i < n

    Initialise j to 0. For j < n

$ma[i][j] = m1[i][j] + m2[i][j]$

        Increment j

    Increment i

Step 3: Print elements of matrix 'ma' using for loop

## Code:

### Main Function

```
#include <stdio.h>

// Function declaration
int n, i, j;
void matrixaddn(int m1[n][n], int m2[n][n]);

// Main function
int main() {

    printf("\nEnter a natural number 'n' for dimensions of the matrices: ");
    scanf("%d", &n);

    // Matrix input from user
    int m1[n][n], m2[n][n];
    printf("\nEnter elements for Matrix 1:\n");
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            scanf("%d", &m1[i][j]);
        }
    }

    printf("\nEnter elements for Matrix 2:\n");
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            scanf("%d", &m2[i][j]);
        }
    }

    matrixaddn(m1, m2);
    printf("\n");
    return 0;
}
```

# Matrix Addition Function

```
// Matrix Addition function
void matrixaddn(int m1[n][n], int m2[n][n]) {
    int ma[n][n];
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            ma[i][j] = m1[i][j] + m2[i][j];
        }
    }

    printf("Matrix Addition:\n");
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            printf("%d ", ma[i][j]);
        }
        printf("\n");
    }
}
```

The screenshot shows the Visual Studio Code editor with a file named `matrixaddn.c` open. The code in the editor is as follows:

```
1 #include <stdio.h>
2
3 // Function declaration
4 int m, i, j;
5 void matrixaddn(int m1[n][n], int m2[n][n]);
6
7 // Main function
8 int main() {
9
10     printf("\nEnter a natural number 'n' for dimensions of the matrices: ");
11     scanf("%d", &n);
12
13     // Matrix input from user
14     int m1[n][n], m2[n][n];
15     printf("\nEnter elements for Matrix 1:\n");
16     for (i = 0; i < n; i++) {
17         for (j = 0; j < n; j++) {
18             scanf("%d", &m1[i][j]);
19         }
20     }
21
22     printf("\nEnter elements for Matrix 2:\n");
23     for (i = 0; i < n; i++) {
24         for (j = 0; j < n; j++) {
25             scanf("%d", &m2[i][j]);
26         }
27     }
28
29     matrixaddn(m1, m2);
30     printf("\n");
31     return 0;
32 }
33
34 // Matrix Addition function
35 void matrixaddn(int m1[n][n], int m2[n][n]) {
36     int ma[n][n];
37     for (i = 0; i < n; i++) {
38         for (j = 0; j < n; j++) {
39             ma[i][j] = m1[i][j] + m2[i][j];
40         }
41     }
42
43     printf("\nMatrix Addition:\n");
44     for (i = 0; i < n; i++) {
45         for (j = 0; j < n; j++) {
46             printf("%d ", ma[i][j]);
47         }
48         printf("\n");
49     }
50 }
```

The terminal output shows the execution of the program for two different matrix sizes:

First run (n=3):

```
PS C:\Users\Vidhi Shah\Desktop\WITC Sem 2\VC C++ Lab\PPS7> gcc -o matrixaddn matrixaddn.c
PS C:\Users\Vidhi Shah\Desktop\WITC Sem 2\VC C++ Lab\PPS7> ./matrixaddn
Enter a natural number 'n' for dimensions of the matrices: 3
Enter elements for Matrix 1:
7 8 1
3 9 0
1 4 2
Enter elements for Matrix 2:
1 0 8
6 0 9
7 3 5
Matrix Addition:
8 8 9
9 9 9
8 7 7
PS C:\Users\Vidhi Shah\Desktop\WITC Sem 2\VC C++ Lab\PPS7> ./matrixaddn
```

Second run (n=2):

```
Enter a natural number 'n' for dimensions of the matrices: 2
Enter elements for Matrix 1:
1 2
3 4
Enter elements for Matrix 2:
5 6
7 8
Matrix Addition:
6 8
10 12
PS C:\Users\Vidhi Shah\Desktop\WITC Sem 2\VC C++ Lab\PPS7>
```