CHAPTER - 5 → Functions & Recursion

Some times our program gets bigger in size and its not possible for a programmer to track which piece of code is doing what function is a way to break our code into chunks so that it is possible for a programmer to reuse them.

what is a function ?
A function is a block of code which performs a particular task.
A function can be reused by the programmer in a given program any number of times

Example and Syntax of a function

```
# include <stdio.h>

void display();  ⇒ function prototype

int main(){
    int a;
    display();  ⇒ function call
    return 0;
}
```

```
void display (){        ⇒ function definition
    printf ("Hi I am display");
}
```

Function prototype ~ Function prototype is a very way to tell the compiler about the function we are going to define in the program. Here void represents that the function is returning nothing.

Function Call ~ Function calls is a way to tell the compiler to Execute the function body at the time the calls is made.
Note that the program Execution Starts from the main function in the sequence the instructions are written.

Function definition ~ This part contains the exact set of instructions which are executed during the function call when a function is called from main(), the main function falls asleep and gets temporarily suspended During this time the control goes to the function being called. when the function body is done Executing main resumes.

Quick Quiz :- write a program with three function
(1) Good morning function which prints "Good Morning"
(2) Good afternoon function which prints "Good Afternoon"
(3) Good night function which prints "Good night"

main() should call all of these in Order 1→2→3

→
```
#include <stdio.h>

void goodmorning (){
    printf ("Good morning\n");
}
void goodafternoon (){
    printf ("Good Afternoon\n");
}
void goodnight (){
    printf ("Good night\n");
}

int main (){
    goodmorning ();
    goodafternoon ();
    goodnight ();

    return 0;
}
```

Important points

→ Execution of a C program from main()
→ A C program can have more than one function
→ Every function gets called directly or indirectly from main()
→ There are two types of functions in C lets talk about them

Types of functions

(1) Library functions → Commonly required functions grouped together in a library file on disk

(2) User defined functions → These are the functions declared and defined by the user.

Why use functions?

(1) To avoid rewriting the same logic again and again
(2) To keep track of what we are doing in a program
(3) To test and check logic independently.

Passing values to functions

We can pass values to a function and can get a value in return from a function.

int Sum (int a, int b)

The above prototype means that Sum is a function which takes values a (of type int) and b (of type int) and returns a value of type int.

Function definition of Sum can be

```
int Sum (int a, int b){
    int c;
    c = a+b;
    return c;
}
```

⇒ a and b are parameters

Now we can call Sum (2,3); from main to get 5 in return ↳ (here 2 & 3 are arguments

int d = Sum (2,3); → d becomes 5

Note

(1) Parameters are the values or variable placeholders in the function definition
Ex:- a & b

(2) Arguments are the actual values passed to the function to make a call
Ex :- 2 & 3.

(3) A function can return only one value at a time

(4) If the passed variable is changed inside the function, the function call doesn't change the value in the calling function

```
int change (int a) {
        a = 77;
        return 0;
}
```

Change is a function which changes a to 77
Now if we call it from main like this

```
int b = 22
change (b);                    ⇒ The value of b
printf ("b is %d", b);           remains 22
                                 ⇒ prints "b is 22"
```

This happens because a copy of b is passed to the change function

Quick Quiz → Use the library function to calculate the area of a square with side a.

```
#include <stdio.h>
int areaSquare (int side) {
        return side * side;
}
```

```
int main (){
    int side;
    printf ("Enter the side length of the
                                Square: ");
    scanf ("%d", &side);

    int area = areaSquare(side);
    printf("The area of the Square is %d \n",
                                area);

    return 0;
}
```

Recursion ~ A function defined in C can call itself This is called Recursion A function calling itself is also called Recursion and 'Recursive function'.

Example of Recursion

A very good Example of recursion is factorials

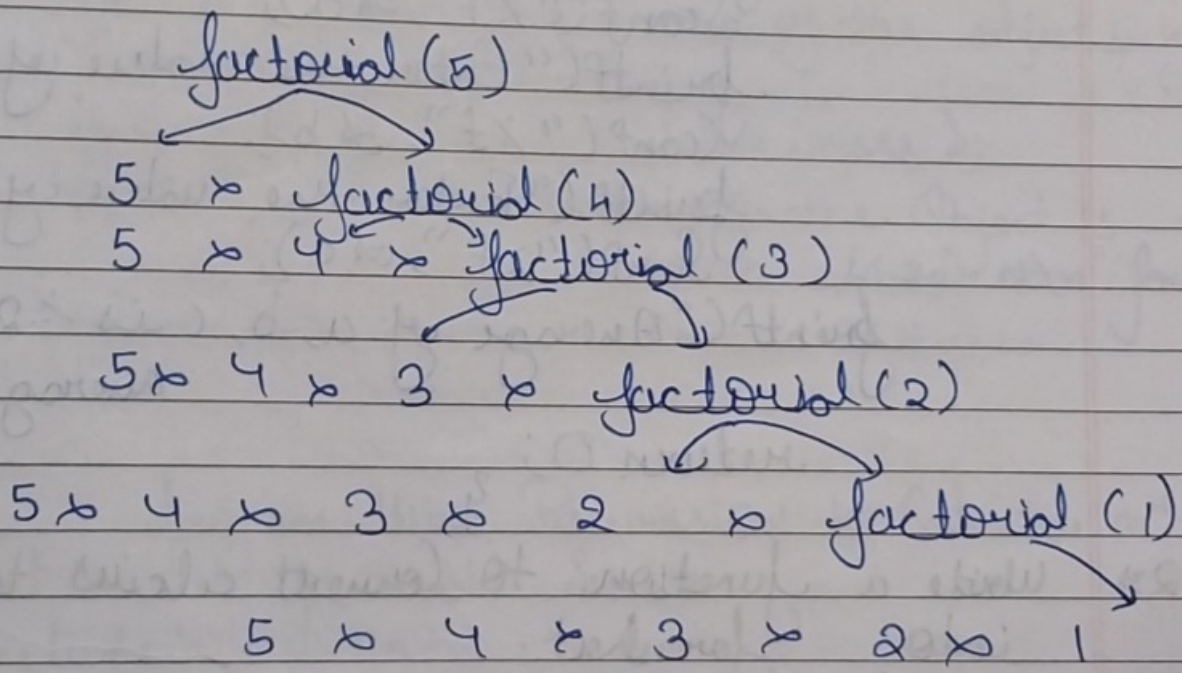
$$\text{factorial}(x) = 1 \times 2 \times 3 \ldots \times x$$
$$\text{factorial}(x) = 1 \times 2 \times 3 \ldots (n-1) \times n$$
$$\text{factorial}(x) = \text{factorial}(n-1) \times n$$

Since we can write factorial of a number in terms of itself, we can program it using recursion

```
int factorial (int x) {
    int f;
    if (x==0 || x==1)
        return 1;
    else
        f = x * (factorial(x-1));
    return f;
}
```

⇒ A program to calculate factorial using recursion

How does it work?



Important Notes :-

(1) Recursion is sometime the most direct way to code an algorithm

(2) The condition which doesn't call the function any further in a recursive function is called as the base condition.

(3) Sometimes due to a mistake made by the programmer a recursive function can keep running without returning resulting in a memory error.

Chapter-5 (Practice Set)

Q1) Write a program using function to find average of three numbers?

```
#include <stdio.h>
float average (float a, float b, float c){
    return (a+b+c)/3.0; }
int main () {
    float a, b, c;
    printf ("Enter the value of a: ");
    scanf("%f", &a);
    printf("Enter the value of b: ");
    scanf("%f", &b);
    printf("Enter the value of c: ");
    scanf("%f", &c);
printf("Average of a, b, c is %.2f \n"
                        average(a,b,c));
    return 0;
}
```

Q2) Write a function to Convert celcius temperature into farenheit.

```
#include <stdio.h>
float temp(float celsius){
    return (celsius * 9.0/5.0)+32.0;
}
int main(){
    float celsius;
    printf("Enter the temprature in celsius: ");
    scanf("%f", &celsius);
    printf("Temprature in fahrenheit: %.2f\n",
                        temp(celsius));
```

return 0;
}

Q3) Write a function to calculate force of attraction on a body of mass m exerted by earth ($g = 9.8 m/s^2$)

```
#include <stdio.h>
float force_calculator(float mass){
    return mass * 9.81; }

int main(){
    float mass;
    printf("Enter the mass of the object in kg: ");
    scanf("%f", &mass);
    float force = force_calculator(mass);
    printf("The force acting on the Object is %.2f N\n", force);
    return 0;
}
```

Q4) Write a program Using Recursion to Calculate n^{th} Element of fibonacci Series

```
#include <stdio.h>
    if (n==0) return 0;
    if (n==1) return 1;
    return Series(n-1) + Series(n-2);
}

int main(){
    int n;
    printf("Enter a number: ");
    scanf("%d", &n);
    Series(n);
    printf("The %d the number in the fibonacci series is : %d\n", n, Series(n));
```

Q5) what will the following line produce in a C program:

printf("%d %d %d\n", a, ++a, a++);

→ 4 4 6
6 5 6
⋮
all of them are correct depends on compiler to compiler

→ It is a logical answer but C behaviour is different in this case.

Q6) Write a recursive function to calculate the sum of first n natural numbers

```
#include <stdio.h>
int f(int n){
  if(n==0) return 0;
  if(n==1) return 1;
  return f(n-1) + (n);}

int main(){
  int n;
  printf("Enter the value of n: ");
  scanf("%d", &n);

  int result = f(n);
  printf("The sum of the first %d
          natural number is: %d\n",
          n, result);
  return 0;
}
```

Q7 Write a program using function to print the following pattern (Given n lines)

```
*
* * *
* * * * *
```

```
#include <stdio.h>
int main(){
    int n;
    printf("Enter the value of n: ");
    scanf("%d", &n);

    for (int i=0; i<=n; i++) {
        for (int j=0; j<2*i+1; j++){
            printf("*");
        }
        printf("\n");
    }

    return 0;
}
```