

NOTE ON RECURSION

1

Recursion is a process by which a function calls itself repeatedly until some specified condition is satisfied. A function calling itself repeatedly to compute a value is called recursive function.

Two conditions must be satisfied to solve a problem recursively –

- i) Problem must be written in recursive form.
- ii) Problem statement must include a stopping condition.

Problem : Factorial of a positive integer number using recursion

Recursive Definition:

$$N! = \begin{cases} 1 & \text{if } N = 0 \\ N * (N - 1)! & \text{if } N > 0 \end{cases}$$

Write a C function to find factorial of a number using **recursion**. Also call that function from main()

```
#include <stdio.h>
```

```
int main()
```

```
{
    int n;                //declaration of a integer variable
    long int fact(int);    //Function prototype
    printf("\n Enter the number : ");
    scanf("%d", &n);
    printf("\n Factorial of %d is %ld", n, fact(n));
    return(0);
}
```

```
long int fact(int n)
```

```
{
    if(n == 0)
        return(1);
    else
        return(n * fact(n-1));
}
```

Problem : Sum of digits of a positive integer number using recursion

Recursive Definition

$$\text{SumDigit}(N) = \begin{cases} N & \text{if } N < 10 \\ N\%10 + \text{SumDigit}(N/10) & \text{if } N \geq 10 \end{cases}$$

Write a C function to find sum of digits of a number using **recursion**. Call that function from main()

```
#include <stdio.h>
```

```
int main()
```

```
{
    int n;                //declaration of a integer variable
    int sumdigit(int);    //Function prototype
    printf("\n Enter the number : ");
    scanf("%d", &n);
    printf("\n Sum of digits of %d is %d", n, sumdigit(n));
    return(0);
}
```

```
int sumdigit(int n)
```

```
{
    if(n < 10)
        return(n);
    else
        return(n%10+sumdigit(n/10));
}
```

Problem : Finding Greatest Common Divisor (GCD) of two positive integers using recursion

Recursive Definition

$$\text{GCD}(a, b) = \begin{cases} a & \text{if } b = 0 \\ \text{GCD}(b, a \% b) & \text{if } b > 0 \end{cases}$$

Write a C function to find GCD of two numbers using **recursion**. Call that function from main()

```
#include <stdio.h>
int main()
{
    int a,b,c;           //declaration of a integer variable
    int gcd(int, int);    //Function prototype
    printf("\n Enter the numbers : ");
    scanf("%d%d", &a,&b);
    c=gcd(a, b);
    printf("\nGreatest Common Divisor of %d and %d is %d", a, b, c);
    return(0);
}

int gcd(int a, int b)
{
    if(b==0)
        return(a);
    else
        return(gcd(b,a%b));
}
```

Problem : Fibonacci series Generation using recursion

Recursive Definition

$$\text{fib}(N) = \begin{cases} N & \text{if } N = 0 \text{ or } 1 \\ \text{fib}(N-2) + \text{fib}(N-1) & \text{if } N > 1 \end{cases}$$

Write a C function to find **n** terms of Fibonacci series using **recursion**. Call that function from main()

```
#include <stdio.h>
int main()
{
    int m,i;           //declaration of a integer variable
    int fib(int);       //Function prototype
    printf("\n How many terms ? ");
    scanf("%d", &n);
    for(i=0; i<n; i++)
        printf("%3d",fib(i));
    return(0);
}

int fib(int n)
{
    if(n == 0 || n == 1)
        return(n);
    else
        return(fib(n-2)+fib(n-1));
}
```

Problem : **Finding X^Y using recursion** where X and Y are two integers

Recursive Definition

$$\text{power}(X, Y) = \begin{cases} 1 & \text{if } Y = 0 \\ x * \text{power}(X, Y-1) & \text{if } Y > 0 \\ 1/X * \text{power}(X, Y+1) & \text{if } Y < 0 \end{cases}$$

Assignment : **Write a C function to find X^Y using recursion.** Also call the function from main().

Problem : **Finding reverse of a positive integer number using recursion.**

Recursive Definition

$$\text{reverse}(N) = \begin{cases} N & \text{if } N < 10 \\ N \% 10 * 10^d + \text{reverse}(N/10) & \text{if } N \geq 10 \end{cases}$$

where d = (No. of digits in N) – 1

Example:

$$\begin{aligned} \text{reverse}(123) &= 123 \% 10 * 10^2 + \text{reverse}(12) \\ &= 123 \% 10 * 10^2 + (12 \% 10 * 10^1 + \text{reverse}(1)) \\ &= 123 \% 10 * 10^2 + (12 \% 10 * 10^1 + 1) \\ &= 123 \% 10 * 10^2 + (21) \\ &= 300 + 21 \\ &= 321 \end{aligned}$$