Recursion

Recursion in computer science is a method where the solution to a problem depends on solutions to smaller instances of the same problem (as opposed to iteration). The approach can be applied to many types of problems, and recursion is one of the central ideas of computer science.

- **Write a C program to find sum of first n natural numbers using recursion.**

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

int sum(int n);

void main()
{
    int num,add;
    printf("Enter a positive integer:\n");
    scanf("%d",&num);
    add=sum(num);
    printf("sum=%d",add);
}
int sum(int n)
{
    if(n==0)
       return n;
    else
       return n+sum(n-1);     /*self call  to function sum() */
}
```

- **Example to calculate GCD or HCF using recursive function.**

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

int hcf(int n1, int n2);

void main()
{
   int n1, n2;
   printf("Enter two positive integers: ");
```

```
   scanf("%d%d", &n1, &n2);
   printf("H.C.F of %d and %d = %d", n1, n2, hcf(n1,n2));
   return 0;
}

int hcf(int n1, int n2)
{
    if (n2!=0)
       return hcf(n2, n1%n2);
    else
       return n1;
}
```

- **Source code to find factorial of a number.**

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

int factorial(int n);

void main()
{
    int n;
    printf("Enter an positive integer: ");
    scanf("%d",&n);
    printf("Factorial of %d = %ld", n, factorial(n));
    return 0;
}

int factorial(int n)
{
    if(n!=1)
     return n*factorial(n-1);
}
```

- **Source Code to calculate power using recursive function**

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

int power(int n1,int n2);

void main()
{
    int base, exp;
```

```
    printf("Enter base number: ");
    scanf("%d",&base);
    printf("Enter power number(positive integer): ");
    scanf("%d",&exp);
    printf("%d^%d = %d", base, exp, power(base, exp));
    return 0;
}

int power(int base,int exp)
{
    if ( exp!=1 )
        return (base*power(base,exp-1));
}
```

- **C Program to find the nth number in Fibonacci series using recursion**

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

    int fibo(int);

    void main()
    {
        int num;
        int result;

        printf("Enter the nth number in fibonacci series: ");
        scanf("%d", &num);
        if (num < 0)
        {
            printf("Fibonacci of negative number is not possible.\n");
        }
        else
        {
            result = fibo(num);
            printf("The %d number in fibonacci series is %d\n", num, result);
        }
    }

    int fibo(int num)
    {
        if (num == 0)
        {
            return 0;
```

```
    }
    else if (num == 1)
    {
        return 1;
    }
    else
    {
        return(fibo(num - 1) + fibo(num - 2));
    }
}
```

- **C Program to Perform Matrix Multiplication using Recursion**

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

void multiply(int, int, int [][10], int, int, int [][10], int [][10]);
void display(int, int, int[][10]);

void main()
{
    int a[10][10], b[10][10], c[10][10] = {0};
    int m1, n1, m2, n2, i, j, k;

    printf("Enter rows and columns for Matrix A respectively: ");
    scanf("%d%d", &m1, &n1);
    printf("Enter rows and columns for Matrix B respectively: ");
    scanf("%d%d", &m2, &n2);
    if (n1 != m2)
    {
        printf("Matrix multiplication not possible.\n");
    }
    else
    {
        printf("Enter elements in Matrix A:\n");
        for (i = 0; i < m1; i++)
        for (j = 0; j < n1; j++)
        {
            scanf("%d", &a[i][j]);
        }
        printf("\nEnter elements in Matrix B:\n");
        for (i = 0; i < m2; i++)
        for (j = 0; j < n2; j++)
        {
            scanf("%d", &b[i][j]);
```

```
        }
        multiply(m1, n1, a, m2, n2, b, c);
    }
    printf("On matrix multiplication of A and B the result is:\n");
    display(m1, n2, c);
}

void multiply (int m1, int n1, int a[10][10], int m2, int n2, int b[10][10], int
c[10][10])
{
    static int i = 0, j = 0, k = 0;

    if (i >= m1)
    {
        return;
    }
    else if (i < m1)
    {
        if (j < n2)
        {
            if (k < n1)
            {
                c[i][j] += a[i][k] * b[k][j];
                k++;
                multiply(m1, n1, a, m2, n2, b, c);
            }
            k = 0;
            j++;
            multiply(m1, n1, a, m2, n2, b, c);
        }
        j = 0;
        i++;
        multiply(m1, n1, a, m2, n2, b, c);
    }
}

void display(int m1, int n2, int c[10][10])
{
    int i, j;

    for (i = 0; i < m1; i++)
    {
        for (j = 0; j < n2; j++)
        {
            printf("%d  ", c[i][j]);
```

```
        }
        printf("\n");
    }
}
```

- **C Program to find Sum of Digits of a Number using Recursion**

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

int sum (int a);

void main()
{
    int num, result;
    printf("Enter the number: ");
    scanf("%d", &num);

    result = sum(num);

    printf("Sum of digits in %d is %d\n", num, result);
}

int sum (int num)
{
    if (num != 0)
    {
        Return (num % 10 + sum (num / 10));
    }
    else
   {
       return 0;
   }
}
```

- **C program to find the reverse of a number using recursion**

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

int rev(int, int);

void main()
{
    int num, result;
    int length = 0, temp;

    printf("Enter an integer number to reverse: ");
    scanf("%d", &num);
    temp = num;
    while (temp != 0)
    {
        length++;
        temp = temp / 10;
    }
    result = rev(num, length);
    printf("The reverse of %d is %d.\n", num, result);
}

int rev(int num, int len)
{
    if (len == 1)
    {
        return num;
    }
    else
    {
        return (((num % 10) * pow(10, len - 1)) + rev(num / 10, --len));
    }
}
```

- **C Program to find whether a Number is Prime or Not using Recursion**

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

int primeno(int, int);

void main()
{
    int num, check;
    printf("Enter a number: ");
    scanf("%d", &num);
    check = primeno(num, num / 2);
    if (check == 1)
    {
        printf("%d is a prime number\n", num);
    }
    else
    {
        printf("%d is not a prime number\n", num);
    }
 }

int primeno (int num, int i)
{
    if (i == 1)
    {
        return 1;
    }
    else
    {
      if (num % i == 0)
      {
        return 0;
      }
      else
      {
        return primeno(num, i - 1);
      }
    }
```

```
}
```

Tail Recursion

A function that calls itself, and doesn't perform any task after function call, is know as **tail recursion**. In tail recursion, we generally call the same function with return statement.

```
// An example of tail recursive function

void print(int n)
{
   if (n < 0)  return;
   printf(" %d ",n);

   // The last executed statement is recursive call

   print(n-1);
}
```

Example of tail recursion in C

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

int factorial (int n)
{
   if ( n < 0)
      return -1;                    /*Wrong value*/
   if (n == 0)
      return 1;                     /*Terminating condition*/
   return (n * factorial (n -1));
}

void main()
{
   int fact=0;
   clrscr();

   fact=factorial(5);
   printf("\n factorial of 5 is %d", fact);
getch();
}
```