Programming for Problem Solving

(ESCG101)

Module V

Recursion:

Recursion is the process of making the function call itself directly or indirectly. A recursive function solves a particular problem by calling a copy of itself and solving smaller sub-problems that sum up the original problems.

Recursion helps to reduce the length of code and make it more understandable. The recursive function uses a LIFO (Last In First Out) structure like a stack. Every recursive call in the program requires extra space in the stack memory.

While using recursion, programmers need to be careful to define an exit condition from the function, otherwise it will go into an infinite loop.

Recursive functions are very useful to solve many mathematical problems, such as calculating the factorial of a number, generating Fibonacci series, etc.

Example:

```
int main ()
{
        ................
        rec ();
        ................
}  /* End of main () */
void rec ()
{
        ................
        rec ();        /* recursive call */
        ................
} /* End of rec () */
```

- Example: (finding factorial using recursion)

```
int fact(int x)
{
        if(x==1)
            return 1;
        else
            f=x * fact (x-1);// fact
is defined in term of fact itself
}
```

Here the function rec () is calling itself inside its own function body, so rec () is a recursive function.

Factorial of a given number using a recursive function:

```
#include <stdio.h>
long int fact(int n);
int main()
{
   int num;
   printf ("Enter a number: ");
   scanf("%d", &num);
   if (num<0)
      printf ("No factorial for negative number\n");
   else
      printf ("Factorial of %d is %ld\n", num, fact(num));
   return 0;
}
long int fact(int n)
{
   if(n <= 1)
      return 1;
   return (n*fact(n - 1));
 }
```

Output:

```
Enter a number: 5
Factorial of 5 is 120
```

Fibonacci series:

Fibonacci series is a sequence of integers that starts with 0 followed by 1, in this sequence the first two terms i.e. 0 and 1 are fixed, and we get the successive terms by summing up their previous last two terms. i.e, the series follows a pattern that each number is equal to the sum of its preceding two numbers.

Mathematically it can be written as : $F(n)=F(n-1)+F(n-2)$
where F is the Fibonacci function having base values $F(0)=0$ and $F(1)=1$

0 1 1 2 3 5 8 13 21 34 . . . . . .
In this series each number is a sum of the previous two numbers.

Program to generate fibonacci series:

```
# include <stdio.h>
int fib(int n);
int main()
{
   int nterms, i;
   printf ("Enter number of terms: ");
   scanf ("%d", &nterms);
   for (i=0; i<nterms; i++)
   {
      printf ("%d\t", fib(i));
   }
   printf("\n");
   return 0;
}
int fib (int n)
{
   if (n==0)
      return 0;
   else if (n==1)
      return 1;
   else
      return (fib(n-1) + fib(n-2));
}
```

Output:

```
Enter number of terms: 8
0   1   1   2   3   5   8  13
```

Program to find GCD (Greatest Common Divisor) using recursion:

```
# include <stdio.h>
int GCD (int a, int b);
int main ()
{
   int x, y, z;
   printf ("Enter two numbers : ");
   scanf ("%d%d", &x,&y);
   z = GCD (x,y);
   printf ("GCD of two numbers is %d", z);
   return 0;
}
int GCD (int a, int b)
{
   if (b==0)
      return a;
   return GCD (b, a%b);
}
```

Output:

```
Enter two numbers : 35 21
GCD of two numbers is 7
```

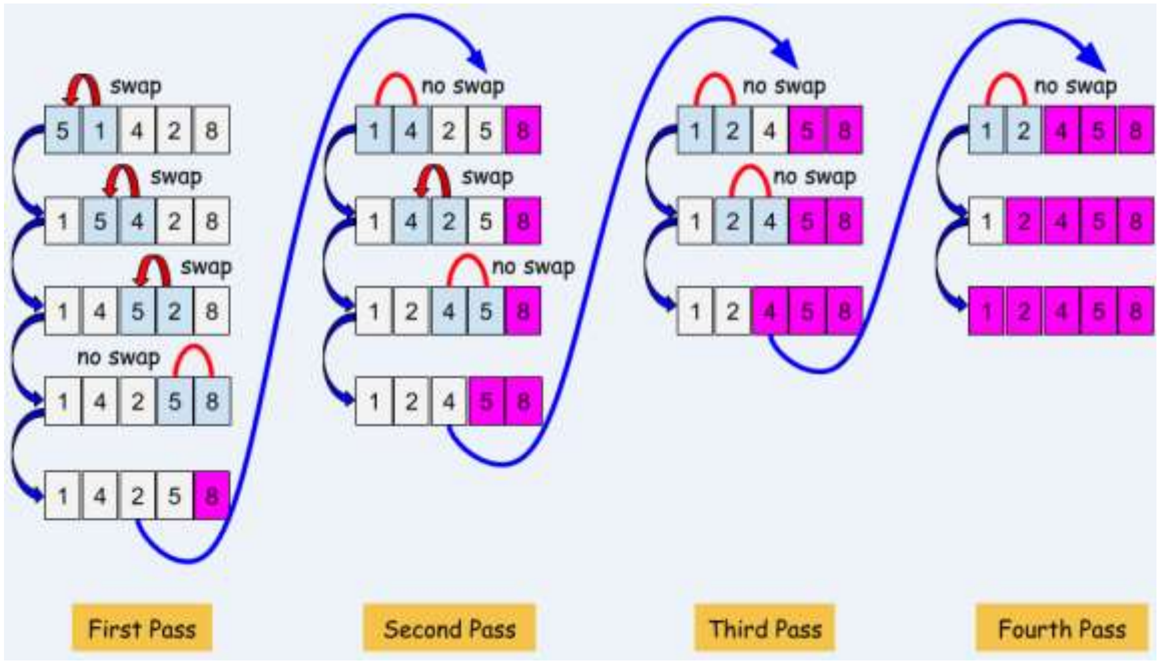
Sorting

- Process of arranging the data in some order (ascending/descending)

- Types:
  - bubble
  - merge
  - quick

Bubble Sort Algorithm, Explain with a program:

Bubble Sort is a simple sorting algorithm commonly used to sort elements in a list or array. It works by repeatedly comparing adjacent elements and swapping them if they are in the wrong order. The algorithm iterates through the list multiple times until no more swaps are needed, resulting in a sorted sequence.



https://www.google.com/search?q=bubble+sort+in+c&sca_esv=576600514&rlz=1C1FKPE_en&hl=en&tbm=isch&sxsrf=AM9HkKnyXT2mf_2lQ0G8Lvfny1ZwhRELJA:1698263952770&source=lnms&sa=X&ved=2ahUKEwiH346d_pGCAxWna2wGHfhWC4wQ_AUoAXoECAEQAw&biw=1024&bih=608&dpr=1.25#imgrc=wmo2mm5TBzOckM&imgdii=7Wz2xhFDUC5bxM

Program to sort an array using Bubble sort algorithm:

```
#include <stdio.h>
void sort (int m, int x[]);
int main()
{
   int i, marks[5] = {40, 90, 73, 81, 35};
   printf ("Marks before sorting\n");
   for (i=0; i<5; i++)
   {
      printf ("%d\t", marks[i]);
   }
   printf ("\n");
   sort (5, marks);
   return 0;
}
void sort (int m, int x[])
{
   int i, j, t;
   for (i=0; i<=m-1; i++)
   {
      for (j=0; j<=m-i; j++)
      {
         if (x[j-1]>=x[j])
         {
            t=x[j-1];
            x[j-1]=x[j];
            x[j]=t;
         }
      }
   }
   printf ("Marks after sorting\n");
   for (i=0; i<5; i++)
   {
      printf ("%d\t", x[i]);
   }
}
```

Output:

```
Marks before sorting
40        90        73        81        35

Marks after sorting
35        40        73        81        90
```

Thank You