Study Material for 2nd Semester Students of Computer Science and BCA
Data Structure: Recursion
Dr Pradip Ghanty
Assistant Professor
Department of Computer Science
Asansol Girls' College

Recursion

Recursion is a process by process by which a function calls itself repeatedly until some specified condition has been satisfied.

The process is used for repetitive computations in which each action is stated in terms of a previous result. Many iterative (i.e. repetitive) problems can be written in this form.

Example: Factorial of a number n

Recursion steps: n! = n * (n-1)! Recursion Termination: 0! = 1

```
1       5! = 5 * 4!
2               4! = 4 * 3!
3                       3! = 3 * 2!
4                               2! = 2 * 1!
5                                       1! = 1 * 0!
6                                               0! = 1

6'      0! = 1
5'      1! = 1 * 0! = 1 * 1 = 1
4'      2! = 2 * 1! = 2 * 1 = 2
3'      3! = 3 * 2! = 3 * 2 = 6
2'      4! = 4 * 3! = 4 * 6 = 24
1'      5! = 5 * 4! = 5 *24 = 120
```

C Program:

```
#include <stdio.h>
/* recursive function for factorial */
long int factorial(int n)
{
   return (n==0) ? 1 : n * factorial(n-1);
}

void main()
{
   int n;
   printf("Enter a number:");
   scanf("%d", &n);
   printf("Factorial of %d is %ld\n", n, factorial(n));
 }
```

Write a recursive function to find the sum of first n natural numbers. Hence write a program to test the function.

```
#include <stdio.h>

/* recursive function to add first n natural numbers */
int addNumbers(int n)
{
        if(n !=0)
            return n + addNumbers(n-1);  /* recursive steps */
        else
            return 0; /* recursion termination */
}

int main()
{
int n;
printf("Enter value of n:");
scanf("%d", &n);
printf("Sum of first %d natural numbers: %d\n", n, addNumbers(n));
return 0;
}
```

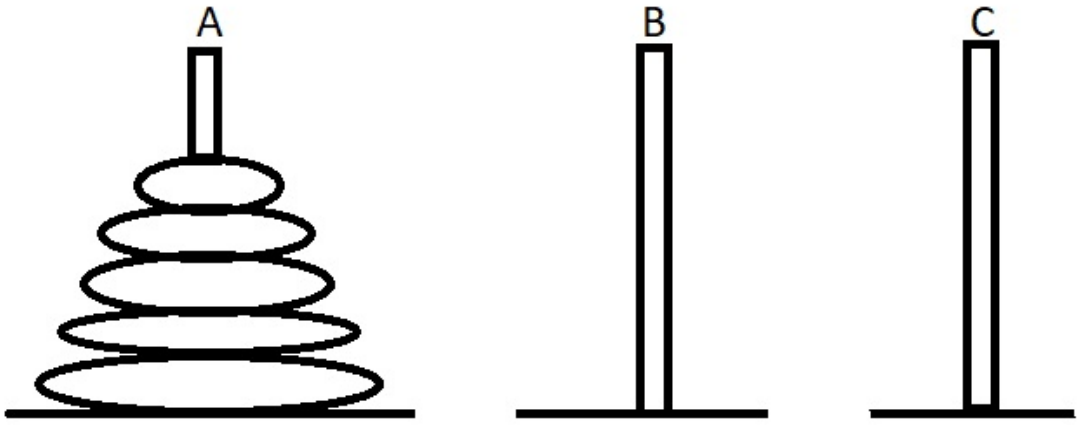
Write a recursive function to find the nth term of a Fibonacci series. Hence write a program to find the sum of first n Fibonacci numbers.

Fibonacci Series: 1, 1, 2, 3, 5, 8....

```
#include <stdio.h>
int fibonacciNumber(int n)
{
        if(n == 1 || n == 2)   /* recursion termination */
                return 1;
        else
                return fibonacciNumber(n-1) + fibonacciNumber(n-2); /* recursive steps */
}

int main()
{
        int n, i, s=0;
        printf("Enter value of n:");
        scanf("%d", &n);
        for(i=1; i<=n; ++i)
                s+=fibonacciNumber(i);
        printf("Sum of first %d Fibonacci numbers: %d\n", n, s);
        return 0;
}
```

The Towers of Hanoi Problem



Problem:
Three Pegs, A, B and C exists. Five disks (let value of n is 5) of different diameters are placed on peg A so that a larger disk is always below a smaller disk. The object is to move the five disks to Peg C, using Peg B as auxiliary. Only the top disk on any peg may be moved to any other pag, and a larger disk may never rest on a smaller one.

The recursive solution to the towers of Hanoi problem as follows:

To move n disks from A to C, using B as auxiliary:

1. If n==1, move the single disk from A to C and stop.
2. Move the top n-1 disks from A to B, using C as auxiliary.
3. Move the remaining disk from A to C.
4. Move the n-1 disks from B to C, using A as auxiliary.

Note: Number of moves required 2^n-1, where n is the number of disks.

C Program:

```
#include <stdio.h>
int cnt = 0;
void towers(int n, char frompeg, char topeg, char auxpeg)
{
   /* if only disk, make the move and return. Recursion termination  */
   if(n == 1)
   {
      printf("\n %d: Move disk %d from peg %c to peg %c\n", ++cnt, n, frompeg, topeg);
      return;
   }
   /* Move top n-1 disks from A to B, using C as auxiliary */
   towers(n-1, frompeg, auxpeg, topeg); /* recursive steps */
```

```
   /* Move remaining disk A to C. Recursion termination .*/
   printf("\n %d: Move disk %d from peg %c to peg %c", ++cnt, n, frompeg, topeg);

   /* Move n-1 disks from B to C using A as auxiliary */
   towers(n-1, auxpeg, topeg, frompeg); /* recursive steps */
}

void main()
{
   int n;
   printf("Enter number of disks:");
   scanf("%d", &n);
   towers(n, 'A', 'C', 'B');
}
```

Output:

```
Enter number of disks:3

 1: Move disk 1 from peg A to peg C

 2: Move disk 2 from peg A to peg B
 3: Move disk 1 from peg C to peg B

 4: Move disk 3 from peg A to peg C
 5: Move disk 1 from peg B to peg A

 6: Move disk 2 from peg B to peg C
 7: Move disk 1 from peg A to peg C
```

Home Work:

1. Write a recursive function to multiply two natural numbers.
2. Write a recursive function to find the sum of the series:
   2 + 4 + 6 + 8 + ... + 2n.