LAB-03 Practice Lab

Using Recursion.

(1) Tower of Hennoi

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

int main()
{
    int n;
    printf ("Enter the no. of disks : ");
    scanf ("%d", &num);
    printf ("The sequence of move as follows
                in Tower of Hanoi is :\n");

    towers(n,'S','T','D');
    return 0;
}

void towers (int n, char src, char temp,
                char dest)
{
    if (num==1)
        { printf ("\n Move disk 1 from
%c to %c", src, dest)
        return;
        }
    towers (n-1, src, dest, temp);
    printf ("\n move disk %d from %c
        to %c \n", n, src, dest);

    towers (n-1, temp, src, dest);

}
```

(2) Factorial.

```
#include <stdio.h>
int fact (int n);

int main()
 {
    int n, f;
    printf("Enter the value of n \n");
    Scanf ("%d", &n);
    printf f=
      f= fact(n)
      printf(" factorial = %d", f);
 }

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

(3) Faibonacci Series:

```
# include < stdio.h>
 int fib (int);

 void main()
 {
   int n, i;
    printf ("Enter the value of n?");
     Scanf (" %d", &n);
     printf (" %d fib num are \n", n);
```

```
for (i=0 ; i<n ; i++)
 {
        printf (" fib (%d) = %d\n", i, fib(i));
 }
}

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

(4) GCD program.

```
# include <stdio.h>
int hcf (int n1, int n2);
int main () {
        int n1, n2;
        printf (" Enter 2 positive integers: ");
        scanf ("%d %d", &n1, &n2);
        printf (" GCD of %d and %d
                is %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;
    }
```

(5) Binary Search.

```
#include <stdio.h>

int binarySearch(int arr[], int l, int r,
                 int x)
{
  if (r >= l) {
    int mid = l + (r - l)/2

    if (arr[mid] == x)
      return mid;

    if (arr[mid] > x)
      return binarySearch(arr, l, mid-1, x);

    return binary Search(arr, mid+1, r, x);
  }

  return -1;
}

int main(void)
{
  int arr[] = {2,3,4,10,40};
  int n = sizeof(arr)/sizeof(arr[0]);
  int x = 10;
  int result = binarySearch(arr, 0, n-1, x);

  (result == -1)? printf("Element not fount")
                : printf("Elemlement found
    at index %d", result);

  return 0;
}
```