Date 12/10/2020

Expt. No. ---------- PRACTICE PROGRAMS Page No. 7

1) WAP in C to find factorial of n using recursion.

```
#include <stdio.h>
int fact (int n)
{
 if (n == 0)
  return 1;
  return n * fact (n-1);
}
int main ()
{
 int n;
printf ("Enter the value of n: \n");
scanf ("%d", &n);
printf (" The factorial of %d = %d \n", n, fact (n));
return 0;
}
```

Teacher's Signature : _____

Output:-

```
Enter the value of n :
7
The factorial of 7 = 5040
```

Date----------

Expt. No.---------- Page No.---8---

2) WAP in C to display n fibonacci numbers using recursion.

```
#include <stdio.h>
int fib (int n)
{
 if (n==0)
 return 0;
 if (n==1)
  return 1;
 return fib (n-1) + fib (n-2);
}
int main ()
{
 int i, n;
printf (" ENTER N :\n");
scanf ("%d", &n);
printf ("%d FIBONACCI NUMBERS ARE :\n", n);
for (i= 0; i<n; i++)
{
 printf (" fib (%d) = %d\n", i, fib(i));
 }
 return 0;
}
```

Teacher's Signature : _____

Output:-

```
Enter N:
9

9 FIBONACCI NUMBERS ARE:-
fib(0) = 0
fib(1) = 1
fib(2) = 1
fib(3) = 2
fib(4) = 3
fib(5) = 5
fib(6) = 8
fib(7) = 13
fib(8) = 21
```

Date----------

Expt. No.---------- Page No. 9

3) Write a C program to find GCD of two nos. using recursion.

```
#include<stdio.h>
int gcd(int m, int n)
{
  if (n==0)
   return m;
  if (m<n)
  return gcd (n,m);
  return gcd (n, m%n);
}
 int main()
{
  int m, n, res;
printf(" ENTER VALUE OF M&N :\n");
scanf(" %d %d", &m &n);
res = gcd (m,n);
printf ("gcd (%d, %d)= %d\n", m,n, res);
 return 0;
 }
```

Teacher's Signature : _____

Output:-

```
ENTER VALUE OF M & N:
24    81
gcd ( 24, 81) = 3
```

Date----------

Expt. No.---------- Page No. 10

4) Write C program to search key element using binary recursive search.

```
#include<stdio.h>
void binary_search(int[], int, int, int);
void bubble_sort(int[], int);
int main()
{
 int key, size, i;
 int list[25];
printf("ENTER LIMIT:");
scanf("%d", &size);
printf("ENTER ELEMENTS:\n");
for(i=0; i<size; i++)
{
  scanf("%d", &list[i]);
}
bubble_sort(list, size);
printf("\n");
printf("ENTER ELEMENT TO SEARCH:\n");
scanf("%d", &key);
binary_search(list, 0, size, key);
 return 0;
}
void bubble_sort(int list[], int size)
{
```

Teacher's Signature : _____

```
int temp, i, j;
for ( i=0; i< size; i++)
{
for (j=0 i; j< size; j++)
 {
   if ( list [i] > list [j])
    {
      temp= list [i];
      list [i] = List [j];
      list [j] = temp;
    }
   }
  }
]
void binary_search (int list [], int lo, int hi, int key)
 {
  int mid;
  if (lo> hi)
  {
   printf (" ELEMENT NOT FOUND \n");
    return;
  }
   mid = (lo+ hi) /2;
   if (list [mid] == key)
   printf (" ELEMENT FOUND \n");
   else if (list [mid] > key)
   binary_ search ( list, lo, mid-1, key);
   else if ( list [mid] <key)
   binary_search (List, mid+1, hi, key);
}
```

Teacher's Signature : _____

Output:-

```
ENTER LIMIT: 4
ENTER ELEMENTS:
87
903
65
11
ENTER ELEMENT TO SEARCH:
903
ELEMENT FOUND


ENTER LIMIT: 2
ENTER ELEMENTS:
97
88
ENTER ELEMENT TO SEARCH:
109
ELEMENT NOT FOUND
```

Date----------

Expt. No.---------- Page No.--12--

5) TOWER OF HANOI

```
# include <stdio.h>
void towers ( int, char, char, char);
int main()
 {
  int num;
printf (" Enter the number of disks :");
scanf ("%d", &num);
printf (" The sequence of moves involved in the Towers of Hanoi are :\n");
towers ( num, 'S', 'T', 'D');
 return 0;
 }
void towers ( int num, char S, char T, char D)
{
 if (num==1)
 {
  printf ("\n Move disk 1 from peg %c to peg %c", S, D);
  return;
  }
 towers ( num-1, S, D, T);
printf (" \n Move disk %d from peg %c to peg %c", num, S, D);
towers (num-1, T, S, D);
}
```

Teacher's Signature : _____

Output:-

```
Enter the number of disks: 4
The sequence of moves involved in the Tower of Hanoi are:
Move disk 1 from peg S to peg T
Move disk 2 from peg S to peg D
Move disk 1 from peg T to peg D
Move disk 3 from peg S to peg T
Move disk 1 from peg D to peg S
Move disk 2 from peg D to peg T
Move disk 1 from peg S to peg T
Move disk 4 from peg S to peg D
Move disk 1 from peg T to peg D
Move disk 2 from peg T to peg S
Move disk 1 from peg D to peg S
Move disk 3 from peg T to peg D
Move disk 1 from peg S to peg T
Move disk 2 from peg S to peg D
```