Batch:P4-1 **Roll No.:16010423076**

Experiment / assignment / tutorial No. 6

Grade: AA / AB / BB / BC / CC / CD /DD

Signature of the Staff In-charge with date

TITLE: Write a program in C to implement user defined functions

AIM:

a) Write a program to find the GCD of two numbers using recursion.
b) Write a program to find the LCM of two numbers by using a) above.

Expected OUTCOME of Experiment:

Design modular programs using functions and the use of structure and union (CO4)

Books/ Journals/ Websites referred:

1. Programming in C, second edition, Pradeep Dey and Manas Ghosh, Oxford University Press.
2. Programming in ANSI C, fifth edition, E Balagurusamy, Tata McGraw Hill.
3. Introduction to programming and problem solving , G. Michael Schneider ,Wiley India edition.

Problem Definition:

1. The program finds the GCD of two numbers using recursion

Example:

Test case 1: Input: 24,28 Output: GCD: 4	Test case 2: Input: 24,25 Output: GCD: 1

2. The program finds the LCM of two numbers using GCD.

Example:

Test case 1: Input: 6,12 Output: LCM: 12	Test case 2: Input: 6,7 Output: LCM: 42

Algorithm:

1)

Ask the user for two positive numbers.
Store these numbers in variables 'num1' and 'num2'.
Define a function 'hcf' that takes two parameters ('num1' and 'num2').
While 'num2' is not 0:
Set 'temp' to 'num2'.
Set 'num2' to 'num1 % num2'.
Set 'num1' to 'temp'.
Return 'num1'.
Call the 'hcf' function with 'num1' and 'num2'.
Print the result as the GCD of the two inputted numbers.

2)

Request the user to enter two positive integers.
Receive and store these integers in variables num1 and num2.
Define a function hcf to calculate the Greatest Common Divisor (GCD) using Euclid's Algorithm.
In the hcf function:
Check if n2 is not equal to 0.
If true, recursively call the hcf function with arguments n2 and n1 % n2.
If false, return n1 as the GCD.
LCM Calculation:
Define a function lcm that takes two parameters (n1 and n2).
In the lcm function:
Calculate the LCM using the formula
(n1*n2)/GCD(n1,n2)
Return the calculated LCM.
In the main function, call the lcm function with num1 and num2.
Print the result obtained from the lcm function, which is the LCM of the two inputted numbers.
Print the LCM of the two inputted positive integers.

Implementation details:

1)

```
#include <stdio.h>

int hcf(int num1,int num2) {
   if (num2 != 0)
      return hcf(num2,num1 % num2);
   else
      return num1;
}

int main() {
   int num1, num2;
   printf("Enter Two +ve numbers : ");
   scanf("%d %d",&num1,&num2);
   printf("%d",hcf(num1,num2));
}
```

2)

```
#include <stdio.h>

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

int lcm(int n1, int n2) {
   return (n1 * n2) / hcf(n1, n2);
}

int main() {
   int num1, num2;
   printf("Enter two +ve integers: ");
   scanf("%d %d", &num1, &num2);
   printf("%d",lcm(num1, num2));
   return 0;
}
```

Output(s):

1)
Test Case 1:

```
Enter Two +ve numbers : 24 28
4
```

Test Case 2:

```
Enter Two +ve numbers : 24 25
1
```

2)
Test Case 1:

```
Enter two +ve integers: 6 12
12
```

Test Case 2:

```
Enter two +ve integers: 6 7
42
```

Conclusion:
In coding these C programs, I gained a practical understanding of implementing recursive algorithms for calculating the Greatest Common Divisor (GCD) and utilizing it to find the Least Common Multiple (LCM). The simplicity and elegance of the recursive approach demonstrated the power of modular design.
This coding exercise reinforced the importance of foundational mathematical concepts in solving real-world problems through programming.

Post Lab Questions

1. Write a C program to find the minimum, maximum and sum of elements in an array using functions.

Input:

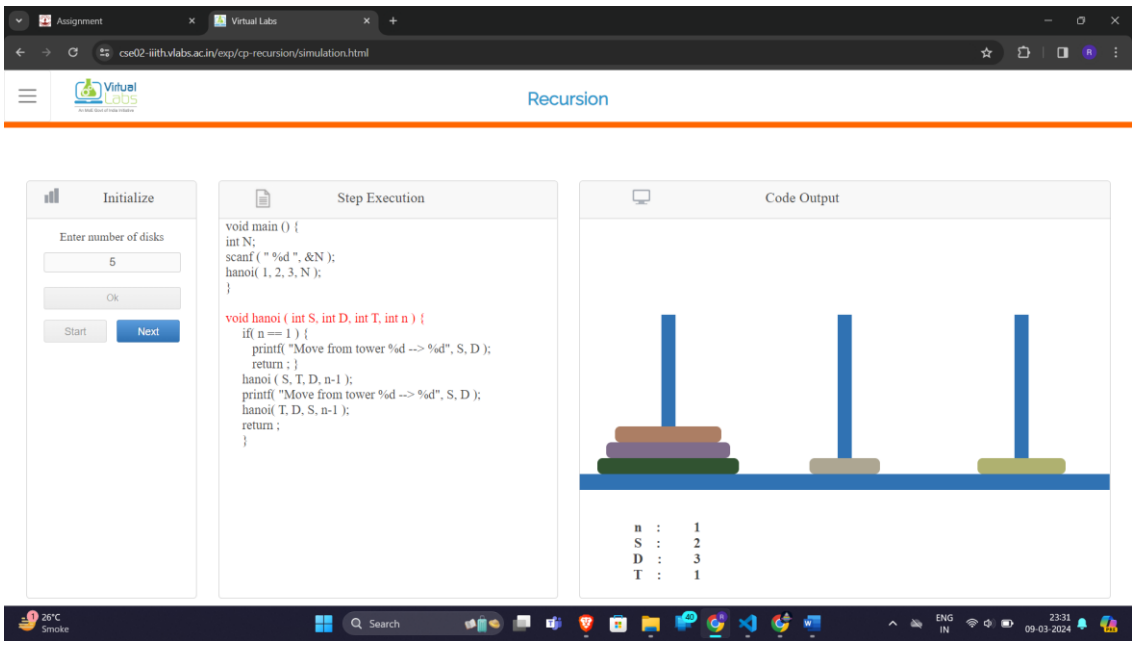
```
#include <stdio.h>

#include <limits.h>


// Sum fn

int findSum(int num[], int n) {

   int sum = 0;

   for (int i = 0; i < n; i++) {

      sum += num[i];

   }

   return sum;

}


// Max fn

int findMax(int num[], int n) {

   int maxi = INT_MIN;

   for (int i = 0; i < n; i++) {

      if (num[i] > maxi) {

         maxi = num[i];

      }

   }

   return maxi;

}
```

```
// Min fn
int findMin(int num[], int n) {
   int min = INT_MAX;
   for (int i = 0; i < n; i++) {
      if (num[i] < min) {
         min = num[i];
      }
   }
   return min;
}

int main() {
   int size;
   scanf("%d", &size);
   int num[100];

   for (int i = 0; i < size; i++) {
      scanf("%d", &num[i]);
   }

   printf("SUM : %d\n", findSum(num, size));
   printf("Maximum : %d\n", findMax(num, size));
   printf("Minimum : %d\n", findMin(num, size));

   return 0;
}
```

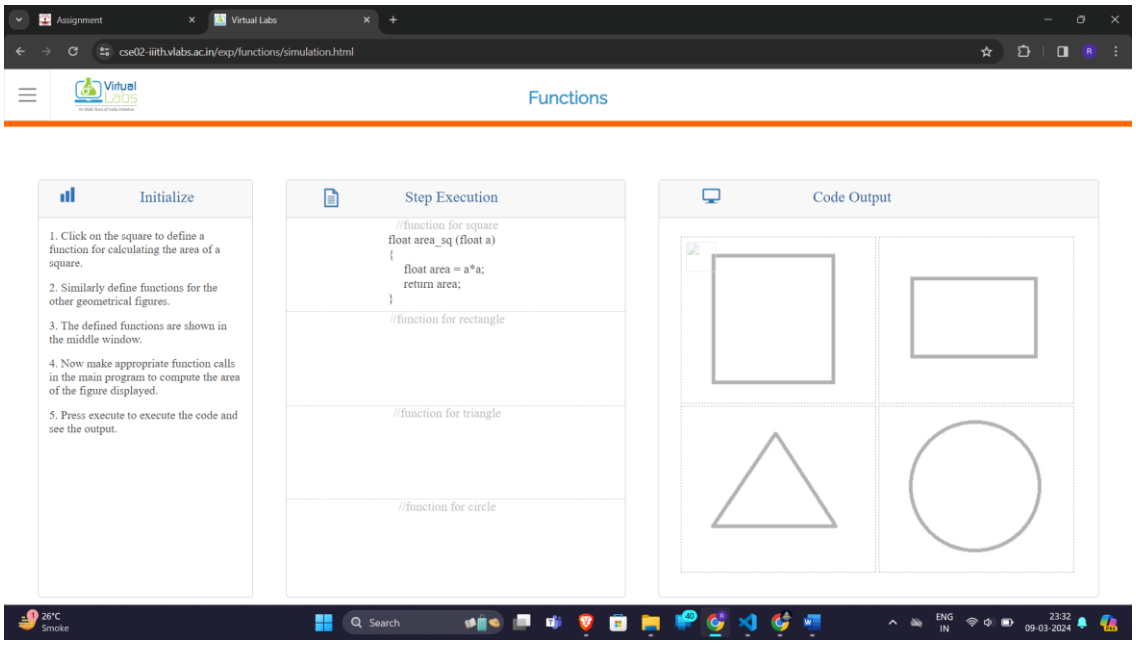
Output:

```
5
12 7 99 1 5
SUM : 124
Maximum : 99
Minimum : 1
```

2. Virtual Lab for functions.
https://cse02-iiith.vlabs.ac.in/exp/cp-recursion/simulation.html



https://cse02-iiith.vlabs.ac.in/exp/functions/simulation.html



Date: _____ **Signature of faculty in-charge**