LAB V

1. Write a C program without using if-else construct that does the following.

   It accepts a sequence of positive integers between 1 and 9 both inclusive from the keyboard. The program will stop accepting input once an integer outside the range is entered. The program will finish by printing the total number multiples of 3 and total number of even integers entered.

   Test data and expected output:

```
Enter integers between 1 & 9 both inclusive, outside range to stop
Enter integer :0
Total no of even integer entered is 0
Total no of multiples of 3 entered is 0

Enter integers between 1 & 9 both inclusive, outside range to stop
Enter integer :2
Enter integer :4
Enter integer :6
Enter integer :9
Enter integer :3
Enter integer :1
Enter integer :2
Enter integer :0
Total no of even integer entered is 4
Total no of multiples of 3 entered is 3
```

2. The equation $f(x) \equiv (1-x)\cos x - \sin x = 0$ has at least one root between $a = 0$ and $b = 1$ since $f(a)f(b) < 0$. The bisection method of finding the root proceeds as follows:

   a. It finds the midpoint $r = (a+b)/2$.

   b. If $f(r) = 0$, then r is the root. If $|b-a|$ is very small less than ϵ, then also we can take r as the root. In either of the cases, our job is done.

   c. If $f(r) \neq 0$ and $f(a)f(r) < 0$, then the root lies between a and r. We assign r to b and go to step a.

   d. If $f(r) \neq 0$ and $f(b)f(r) < 0$, then the root lies between r and b. We assign r to a and go to step a.

   e. If the number of iterations is high, we may stop the process with appropriate message.

   Write the following functions with the specifications mentioned.
   1. Function **func** takes a real number x as argument and returns the value of $f(x)$.
   2. Function **cbracket** takes two real numbers a and b as arguments and returns 1 if at least one real root of $f(x)$ lies between a and b, and 0 otherwise.
   3. Function **rootb** that takes three real numbers a, b, eps and an integer $Nmax$ as arguments. This function returns the root of $f(x) = 0$ using bisection method. If the number of iteration is more than $Nmax$ then the function terminates with appropriate message.

   Write a C program using the above functions. This program accepts a, b, eps and $Nmax$ from the keyboard and prints out the root (if any).

   Test data and expected output:

```
Enter eps and Nmax :1.e-6 20
Enter a, b :0 3
Root must be bracketed

Enter eps and Nmax :1.e-6 10
Enter a, b :0 2
Increase the max iteration

Enter eps and Nmax :1.e-6 50
Enter a, b :0 2
Root = 0.479731
```

3. The greatest common divisor (GCD) of two integers (of which at least one is nonzero) is the largest positive integer that divides the numbers. Write a C non-recursive function **nrgcd** that accepts two integers (assume that both are non-negative and at least one of them is nonzero) and returns their GCD.

   Write a C program (that includes the function **nrgcd**) which accepts two integers. It terminates with appropriate message if both the integers are zero, otherwise it prints their GCD.

   Test data and expected output:

```
Enter two integers: 0 0
At least one number must be nonzero

Enter two integers: 0 -4
GCD of 0 and -4 is 4

Enter two integers: 12 8
GCD of 12 and 8 is 4
```

4. Implement the previous problem with a recursive function **rgcd** that accepts two integers (assume that both are non-negative and at least one of them is nonzero) and returns their GCD.

5. The least common multiple (LCM) of two integers a and b is the smallest positive integer that is divisible by both a and b. If either a or b is 0, LCM of a and b is undefined. Write a C function **lcm** that accepts two integers and returns -1 if either of the integers is zero, otherwise it returns their LCM.

   Write a C program (that includes the function **lcm**) which accepts two integers and prints their LCM.

   Test data and expected output:

```
Enter two integers: 0 5
Both a & b must be nonzero

Enter two integers: 9 12
LCM of 9 and 12 is 36
```

```
Enter two integers: -9 12
LCM of -9 and 12 is 36
```

6. Write a recursive C function **rsumd** that accepts an integer and returns the sum of its digits. Write a C program (that includes the function **rsumd**) which accepts a non-negative integer and prints the sum of its digits.

   Test data and expected output:

```
Enter a non-negative integer: 0
Sum of digits of 0 is 0

Enter a non-negative integer: -4
Number must be non-negative: enter again:234567
Sum of digits of 234567 is 27
```

7. The Tower of Hanoi problem using recursion is listed below. Copy it and run the program for small value of n.

```
#include <stdio.h>
void Tower_of_Hanoi(int,char,char,char);

int main()
{
    int n;
    printf("Enter number of disks :");
    scanf("%d",&n);
    if(n<1)
       {
        printf("Number of disk must be positive\n");
        return 0;
       }

    //Move n disk from stick A  to stick C using stick B
    Tower_of_Hanoi(n, 'A', 'C', 'B');
    return 0;
}

void Tower_of_Hanoi(int n, char from_stk, char to_stk, char help_stk)
{
    if (n == 1)
    {
        printf("Move disk %d from stick %c to stick %c\n",n,from_stk, to_stk);
        return;
    }
    //Move n-1 disk from from_stk to help_stk using to_stk
    Tower_of_Hanoi(n-1,from_stk,help_stk,to_stk);

    //Move n-th disk from from_stk to to_stk
    printf("Move disk %d from stick %c to stick %c\n",n,from_stk,to_stk);

    //Move n-1 disk from help_stk to to_stk using from_stk
```

```
    Tower_of_Hanoi(n-1,help_stk, to_stk, from_stk);
}
```