# PSG COLLEGE OF TECHNOLOGY, COIMBATORE -641004 DEPARTMENT OF COMPUTER APPLICATIONS I SEMESTER MCA
# 23MX16 C PROGRAMMING LABORATORY
# PROBLEM SHEET 2 - WEEK 2

**MITHULESH N [D25MX338]**

1. **Write a C program that accepts a positive integer from the keyboard. If the input is invalid, it stops with appropriate message. For a valid input, it determines the first and last digits of the number. Further, it checks whether the first digit or the last digit is multiple of the other with appropriate message.**

```
#include<stdio.h>
#include<conio.h>
int main()
{
   int n, first, last, temp;
   printf("Enter a positive number: ");
   scanf("%d",&n);
   if(n<=0)
      {
      printf("invalid input.!!");
      return 0;
      }
   last= n%10;
   temp= n;
   while (temp>= 10)
      {
      temp/=10;
      }
   first= temp;
   if(last!=0 && first% last==0)
      {
      printf("1st digit is a multiple of the last digit ");
      }
   else if (first!=0 && last% first==0){
      printf("Last digit is a multiple of the 1st digit ");
   }
```

```
   else{
      printf("Neither digit is a multiple of the other");
   }
   return 0;
}
```

**2. 2. Write a C program that reads two real numbers (from keyboard) representing the x and y coordinates of a point in the Cartesian plane. It then checks whether the point lies inside, outside or on a circle of radius 5 with centre at the origin. Finally, it prints appropriate message too.**

```
#include<stdio.h>
#include<math.h>
int main()
{
   float x,y,dist;
   printf("Enter the coordinates: ");
   scanf("%f%f", &x, &y);
   dist= sqrt(x*x +y*y);
   if(dist < 5){
   printf("The Point lies inside the Circle.!!");
   }
   else if(dist== 5){
      printf("The Point lies on the circle.!!");
   }
   else{
      printf("The Point lies outside the circle.!!");
   }
      return 0;
}
```

**3. Write a C program that accepts a positive integer (from the keyboard). If the input is invalid, it stops after printing the message Invalid input. For a valid input, it then computes and Prints out the sum 1 · n + 2 · (n - 1) + 3 · (n - 2) + · · · · · · + (n - 1) · 2 + n · 1**

```
#include<stdio.h>
int main()
{
   int n, i ,sum =0;
   printf("Enter the Positive number: ");
   scanf("%d",&n);
```

```
   if(n<=0)
   {
      printf("Invalid input!!");
      return 0;
   }
   for(i=1; i<=n; i++)
   {
      sum+= i*(n-i+1);
   }
   printf("The sum is: %d",sum);
   return 0;
}
```

**4. Write a C program that accepts a three digit positive integer from the keyboard. If the input is invalid, it stops after printing the message Invalid input. For a valid input, it then checks whether the sum of the digits is equal to the product of the digits. Finally, it prints appropriate message too. (Example: Typical input: 123**
**Typical output: The sum of the digits is equal to the product of the digits**

```
#include<stdio.h>
int main()
{
   int n,d1,d2,d3,sum,pro;
   printf("Enter a 3 digit Number to process: ");
   scanf("%d",&n);
   if(n< 100 && n>999)
   {
      printf("Invalid number!!");
      return 0;
   }
   d1= n/100;
   d2= (n/10)%10;
   d3= n%10;
   sum= d1+d2+d3;
   pro= d1*d2*d3;
   if(sum==pro)
   {
      printf("The Sum & Product of the number is Equal!!");
   }
   else
   {
      printf("The sum & product of the numbers is not equal!!");
   }
   return 0;
}
```

**5. Write a C program that accepts a three digit positive integer from the keyboard. If the input is invalid, it stops after printing the message Invalid input. For a valid input, it then checks whether the sum of the first and the last digits is equal to the middle digit. Finally, it prints appropriate message too.**

```
#include<stdio.h>
int main()
{
   int n,d1,d2,d3;
```

```
   printf("Enter the 3 digit value: ");
   scanf("%d",&n);
   if(n<100 || n>999)
   {
      printf("Invalid Digit");
      return 0;
   }
   d1= n/100;
   d2= (n/10)%10;
   d3= n%10;
   if(d1+d3 == d2)
   {
      printf("The sum of first digit and last digit = Middle digit ");
   }
   else
   {
      printf("The sum of first digit and last digit != Middle digit ");
   }
   return 0;
}
```

**6. The Bessel function of the first kind of order zero is defined by Write a C program that accepts real $x$ from the keyboard. Then it calculates and prints out the value of $J_0(x)$ using the first 20 terms only.**

```
#include <stdio.h>
int main() {
   int N, sum = 0, expected, duplicate_minus_missing;

   printf("Enter the value of N: ");
   scanf("%d", &N);

   int arr[N];


   printf("Enter %d elements:\n", N);
   for(int i = 0; i < N; i++) {
      scanf("%d", &arr[i]);
      sum += arr[i];
   }

   expected = (N * (N - 1)) / 2;

   duplicate_minus_missing = sum - expected;

   printf("Duplicate - Missing = %d (Use this for duplicate detection)\n", duplicate_minus_missing);

   return 0;
}
```

**7. Find the sum of the series X+XX+XXX+ ….. upto n terms. Where X and n are user inputs.**

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

int main() {
   int X, n;
   long long term = 0, sum = 0;
```

```
        printf("Enter the value of X (a single digit): ");
        scanf("%d", &X);

        printf("Enter the number of terms (n): ");
        scanf("%d", &n);

        for(int i = 0; i < n; i++) {
           term = term * 10 + X;
           sum += term;
        }

        printf("Sum of the series = %d\n", sum);

        return 0;
     }
```

**8. Write C program to print a Magic square for given size 'n'.**

```
#include <stdio.h>
void generateMagicSquare(int n) {
   if (n % 2 == 0) {
      printf("Magic square is only possible for odd values. \n");
      return;
   }

   int magic[n][n];

   for (int i = 0; i < n; i++)
      for (int j = 0; j < n; j++)
         magic[i][j] = 0;

   int number = 1;
   int i = 0, j = n / 2;

   while (number <= n * n) {
      magic[i][j] = number;

      number++;
      int newi = (i - 1 + n) % n;
      int newj = (j + 1) % n;

      if (magic[newi][newj] != 0)
         i = (i + 1) % n;
      else {
         i = newi;
         j = newj;
      }
   }

   printf("\nMagic Square of size %d:\n\n", n);
   for (i = 0; i < n; i++) {
      for (j = 0; j < n; j++)
         printf("%4d", magic[i][j]);
      printf("\n");
   }
}
```

```
int main() {
   int n;

   printf("Enter an odd number (n): ");
   scanf("%d", &n);

   generateMagicSquare(n);

   return 0;
}
```

**9. Write and demonstrate a C program to find all fractions with two-digit numerators and denominators for which "Sleepy's" Technique work correctly**

**10. Consider an array of numbers from 1 to N . In this array, one of the numbers gets duplicated and one is missing. Write a C program to find out the duplicated number. Condition: Using only one loop and without any extra memory.**

```
#include <stdio.h>

int main() {
   int n;
   printf("Enter the size of the array (N): ");
   scanf("%d", &n);

   int arr[n];
   printf("Enter %d elements (array with one duplicate and one missing number):\n", n);
   for (int i = 0; i < n; i++) {
      scanf("%d", &arr[i]);
   }

   long long sum_arr = 0, sum_sq_arr = 0;
   long long sum_n = (long long)n * (n + 1) / 2;
   long long sum_sq_n = (long long)n * (n + 1) * (2 * n + 1) / 6;

   for (int i = 0; i < n; i++) {
      sum_arr += arr[i];
      sum_sq_arr += (long long)arr[i] * arr[i];
   }

   long long diff = sum_arr - sum_n;
   long long diff_sq = sum_sq_arr - sum_sq_n;

   if (diff == 0) {
      printf("No duplicate or missing number detected.\n");
      return 0;
   }

   long long sum_DM = diff_sq / diff;          // D + M

   int D = (diff + sum_DM) / 2;
   int M = (sum_DM - diff) / 2;

   printf("Duplicated number = %d\n", D);
```

```
    printf("Missing number = %d\n", M);

    return 0;
}
```