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

Tino Britty J (D25MX316)

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.

```c
#include <stdio.h>


int main() {

   int n, first, last;

   printf("Enter a positive integer: ");

   if (scanf("%d", &n) != 1 || n <= 0) {

       printf("Invalid input\n");

       return 0;

   }


   last = n % 10;

   while (n >= 10)

       n /= 10;

   first = n;


   if (first % last == 0 || last % first == 0)

       printf("One digit is a multiple of the other\n");

   else

       printf("Digits are not multiples of each other\n");
   return 0;

}
```

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. (Example: Typical input: 2.5 3.0 Typical output: The point lies inside the circle Typical input: 1.9 4.8 Typical output: The point lies outside the circle Typical input: 3.0 4.0 Typical output: The point lies on the circle)

```
#include <stdio.h>

#include <math.h>


int main() {

   float x, y;

   printf("Enter coordinates (x y): ");

   scanf("%f %f", &x, &y);


   float distance = sqrt(x * x + y * y);


   if (distance < 5)

       printf("The point lies inside the circle\n");

   else if (distance > 5)

       printf("The point lies outside the circle\n");

   else

       printf("The point lies on the circle\n");


   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

$1 \cdot n + 2 \cdot (n - 1) + 3 \cdot (n - 2) + \cdots\cdots + (n - 1) \cdot 2 + n \cdot 1$

```c
#include <stdio.h>


int main() {

   int n, sum = 0;

   printf("Enter a positive integer: ");

   if (scanf("%d", &n) != 1 || n <= 0) {

       printf("Invalid input\n");

       return 0;

   }


   for (int i = 1; i <= n; i++) {

       sum += i * (n - i + 1);

   }

   printf("Sum = %d\n", 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 Typical input: 121 Typical output: The sum of the digits is NOT equal to the product of the digits)

```c
#include <stdio.h>


int main() {
   int n, d1, d2, d3;
   printf("Enter a three-digit integer: ");
   scanf("%d", &n);

   if (n < 100 || n > 999) {
       printf("Invalid input\n");
       return 0;
   }

   d1 = n / 100;
   d2 = (n / 10) % 10;
   d3 = n % 10;

   if ((d1 + d2 + d3) == (d1 * d2 * d3))
       printf("The sum of the digits is equal to the product of the digits\n");
   else
       printf("The sum of the digits is NOT equal to the product of the digits\n");

   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 a three-digit integer: ");

   scanf("%d", &n);


   if (n < 100 || n > 999) {

       printf("Invalid input\n");

       return 0;

   }


   d1 = n / 100;

   d2 = (n / 10) % 10;

   d3 = n % 10;


   if (d1 + d3 == d2)

       printf("The sum of first and last digits equals the middle 
digit\n");

   else

       printf("The sum of first and last digits does NOT equal the 
middle digit\n");
```

```
   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 J0(x) using the first 20 terms only.

```
#include <stdio.h>

#include <math.h>


int main() {

   double x, sum = 0.0;

   printf("Enter value of x: ");

   scanf("%lf", &x);


   for (int k = 0; k < 20; k++) {

       double term = pow(-1, k) * pow(x / 2.0, 2 * k) / (tgamma(k +
1) * tgamma(k + 1));

       sum += term;

   }

   printf("J0(%.2lf) ≈ %.6lf\n", x, sum);


   return 0;

}
```

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

```c
#include <stdio.h>


int main() {

   int X, n, term = 0, sum = 0;

   printf("Enter X and n: ");

   scanf("%d %d", &X, &n);


   for (int i = 1; i <= n; i++) {

       term = term * 10 + X;   // build number like X, XX, XXX

       sum += term;

   }

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


   return 0;

}
```

.

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

```c
#include <stdio.h>


int main() {

   int n;

   printf("Enter size of magic square (odd n): ");

   scanf("%d", &n);
```

```
    if (n % 2 == 0 || n <= 0) {
        printf("Invalid input\n");
        return 0;
    }

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

    int i = 0, j = n / 2;
    for (int num = 1; num <= n * n; num++) {
        magic[i][j] = num;
        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("Magic Square:\n");
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++)
            printf("%4d", magic[i][j]);
        printf("\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 i.e. Sleepy’s Technique 26 2 ----- = ----- 65 / 5 Fractions where Sleepy's Technique works: 16/64 = 1/4 19/95 = 1/5 26/65 = 2/5 49/98 = 4/8

```
#include <stdio.h>


int main() {

   int num, den;

   printf("Fractions where Sleepy's Technique works:\n");


   for (num = 10; num < 100; num++) {

       for (den = num + 1; den < 100; den++) {

           int n1 = num / 10, n2 = num % 10;

           int d1 = den / 10, d2 = den % 10;


           if (n2 == d1 && d2 != 0 && (num * 1.0 / den == n1 * 1.0 /
d2)) {

               printf("%d/%d = %d/%d\n", num, den, n1, d2);

           }

       }

   }


   return 0;

}
```

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.

```c
#include <stdio.h>


int main() {

   int N;

   printf("Enter N: ");

   scanf("%d", &N);


   int arr[N];

   printf("Enter %d numbers (1 to N with one duplicate):\n", N);

   for (int i = 0; i < N; i++)

       scanf("%d", &arr[i]);


   int expectedSum = N * (N + 1) / 2;

   int actualSum = 0;

   for (int i = 0; i < N; i++)

       actualSum += arr[i];


   int duplicate = actualSum - expectedSum;

   printf("Duplicate number is %d\n", duplicate);


   return 0;

}
```