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.

CODE:

#include <stdio.h>

int main() {

int n;

printf("Enter a positive number: ");

scanf("%d", &n);

if (n <= 0) {

printf("Invalid input! Please enter a positive integer\n");

return 0;

}

int lastDigit = n % 10;

int firstDigit = n;

while (firstDigit >= 10) {

firstDigit /= 10;

}

printf("First digit: %d\n", firstDigit);

printf("Last digit: %d\n", lastDigit);

if (firstDigit == 0 || lastDigit == 0) {

printf("Cannot check for zero as one of the digits is zero\n");

}

else if (firstDigit % lastDigit == 0) {

printf("First digit is a multiple of last digit\n");

}

else if (lastDigit % firstDigit == 0) {

printf("Last digit is a multiple of first digit\n");

}

else {

printf("Neither digit is a multiple of the other\n");

}

return 0;

}

1. 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)

**CODE:**

#include <stdio.h>

int main() {

float x,y;

float dis\_sq;

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

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

dis\_sq = x\*x+y\*y;

if(dis\_sq==25.0){

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

}

else if(dis\_sq < 25.0){

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

}

else{

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

}

return 0;

}

1. 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

**CODE:**

**#include<stdio.h>**

**int main(){**

**int n,i,term,sum;**

**printf("Enter input:");**

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

**if(n<=0){**

**printf("Invalid input!!");**

**return 0;**

**}**

**sum=0;**

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

**term = i \* (n-i+1);**

**sum = sum + term;**

**}**

**printf("Sum =%d",sum);**

**return 0;**

**}**

1. 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)

**CODE:**

**#include <stdio.h>**

**int main(){**

**int n,digit1,digit2,digit3,sum,product;**

**printf("Enter three digit positive integer:");**

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

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

**printf("Invalid!!!");**

**return 0;**

**}**

**digit1=n/100;**

**digit2=(n/10)%10;**

**digit3=n%10;**

**sum = digit1+digit2+digit3;**

**product=digit1\*digit2\*digit3;**

**printf("Sum of the digits%d:\n",sum);**

**printf("Product of the digits%d:\n",product);**

**if(sum==product){**

**printf("The sum of digits is equal to product of digits:");**

**}**

**else{**

**printf("The sum of digits is Not equal to product of digits:");**

**}**

**return 0;**

**}**

1. 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.

**CODE:**

#include<stdio.h>

int main(){

int n,first,middle,last;

printf("Enter a three digit number:");

scanf("%d",&n);

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

printf("Invalid!!");

return 0;

}

first=n/100;

middle=(n/10)%10;

last=n%10;

printf("First digit:%d\n",first);

printf("Middle digit:%d\n",middle);

printf("Last digit:%d\n",last);

if(first + last == middle){

printf("The sum of first and last digit is equal to middle:");

}

else {

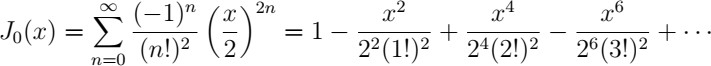
printf("The sum of first and last digit is not equal to middle digit:");

}

return 0;

}

1. 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() {

double x, J0;

printf("Enter value of x: ");

scanf("%lf", &x);

J0 = 1.0;

double fact = 1.0;

double power = 1.0;

double x2 = (x/2.0) \* (x/2.0);

for (int i = 1; i < 20; i++) {

fact \*= i;

power \*= x2;

double term = ( (i % 2 == 0) ? 1 : -1 ) \* (power / (fact \* fact));

J0 += term;

}

printf("J0(%.4f) ≈ %.10f\n", x, J0);

return 0;

}

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

Code:

#include <stdio.h>

int main() {

int X , n ,i;

int sum=0 , term=0;

printf("Enter no of X:");

scanf("%d",&X);

printf("Enter no of n:");

scanf("%d",&n);

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

term=term \* 10 + X;

sum+=term;

}

printf("Sum of series:%d:",sum);

return 0;

}

1. Write C program to print a Magic square for given size ‘n’.

#include <stdio.h>

int main() {

int n,ans,constant;

printf("Enter size of n:");

scanf("%d",&n);

if(n%2==0){

printf("Magic square is only possible for odd numbers:");

return 0;

}

int magicSq[n][n];

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

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

magicSq[i][j]=0;

}

}

int i=0;

int j=n/2;

for(int num=1;num<=n\*n;num++){

magicSq[i][j]=num;

int new\_i=(i-1+n)%n;

int new\_j=(j+1)%n;

if(magicSq[new\_i][new\_j]!=0){

i=(i+1)%n;

}

else{

i=new\_i;

j=new\_j;

}

}

printf("magicSq:%d\n",n);

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

for(int y=0;y<n;y++)

printf("%2d",magicSq[x][y]);

printf("\n");

}

constant=n\*(n\*n+1)/2;

printf("The magic constant is:%d",constant);

return 0;

}

1. 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

Code:

#include <stdio.h>

int main() {

int num , den ;

printf("Sleepy Techniques:\n");

printf("Enter value for numerator:");

scanf("%d",&num);

printf("Enter value for denominator:");

scanf("%d",&den);

if(num < 10 || num > 99 || den < 10 || den > 99){

printf("Both numerator and denominator must be two-digits:");

return 0;

}

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

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

if(n2==0 && d2==0){

printf("Trivial solution is should not be zero in Both numerator & Denominator:");

return 0;

}

int valid = 0;

if(n2==d1 && d2!=0){

if(num \* d2 == den \* n1){

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

valid = 1;

}

}

if(n1==d2 && d1!=0){

if(num \* d2 == den \* n1){

printf("%d/%d works:%d/%d\n",num,den,n2,d1);

valid = 1;

}

}

if (!valid) {

printf("%d/%d does NOT work with Sleepy's technique.\n", num, den);

}

return 0;

}

1. 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 no of values:");

scanf("%d",&n);

int arr[n];

printf("Enter elements in array(from to %d with one missing and one duplicate:\n");

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

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

}

int sumN = n \* (n+1) / 2;

int sqSumN = n \* (n+1) \* (2\*n+1) / 6;

int sum = 0 , sqSum = 0;

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

sum += arr[i];

sqSum += arr[i] \* arr[i];

}

int diff = sum - sumN ;

int sqDiff = sqSum - sqSumN;

int dupMiss = sqDiff / diff ;

int duplicate = (diff + dupMiss) / 2;

int missing = duplicate - diff;

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

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

return 0;

}

Deadline: 26.08.2025