practical:- 1

Aim :- wap display prime number between intervale using function and Armstrong number using user define function.

Input:-

#include <stdio.h>

#include <math.h>

int checkPrimeNumber(int n);

int checkArmstrongNumber(int n);

int main()

{

int n, a;

printf("Enter a positive integer: ");

scanf("%d", &n);

// Check prime number

a = checkPrimeNumber(n);

if (a == 1)

printf("%d is a prime number.\n", n);

else

printf("%d is not a prime number.\n", n);

// Check Armstrong number

a = checkArmstrongNumber(n);

if (a == 1)

printf("%d is an Armstrong number.", n);

else

printf("%d is not an Armstrong number.",n);

return 0;

}

int checkPrimeNumber(int n)

{

int i, a = 1;

for(i=2; i<=n/2; ++i)

{

// condition for non-prime number

if(n%i == 0)

{

a = 0;

break;

}

}

return a;

}

int checkArmstrongNumber(int number)

{

int originalNumber, remainder, result = 0, n = 0, a;

originalNumber = number;

while (originalNumber != 0)

{

originalNumber /= 10;

++n;

}

originalNumber = number;

while (originalNumber != 0)

{

remainder = originalNumber%10;

result += pow(remainder, n);

originalNumber /= 10;

}

// condition for Armstrong number

if(result == number)

a = 1;

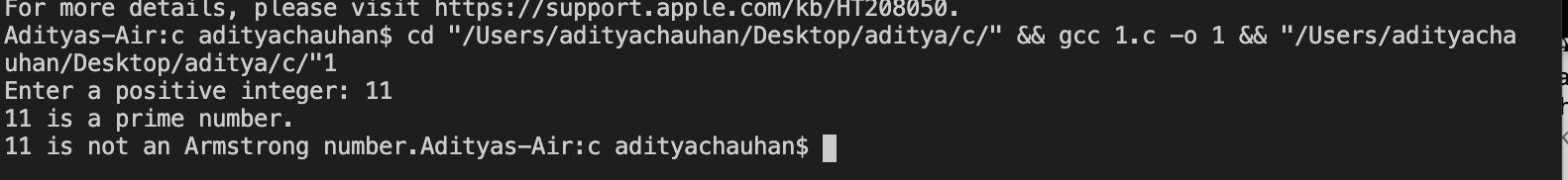
else

a = 0;

return a;

}

Output :-



practical:- 2

Aim :-wap to connect decimal number to binary numbers using the function.

Input:-

#include <stdio.h>

#include <math.h>

long decimalToBinary(int decimalnum)

{

long binarynum = 0;

int rem, temp = 1;

while (decimalnum!=0)

{

rem = decimalnum%2;

decimalnum = decimalnum / 2;

binarynum = binarynum + rem\*temp;

temp = temp \* 10;

}

return binarynum;

}

int main()

{

int decimalnum;

printf("Enter a Decimal Number: ");

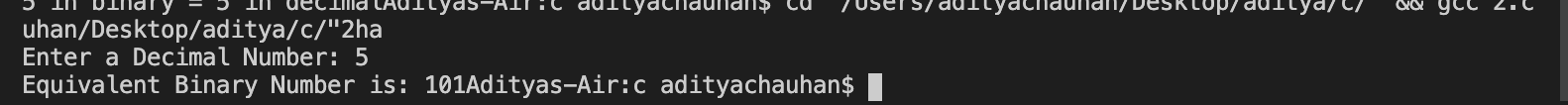
scanf("%d", &decimalnum);

printf("Equivalent Binary Number is: %ld", decimalToBinary(decimalnum));

return 0;

}

Output :-



practical:- 3

Aim :-write a c program to find the some of the natural number using recursion.

Input:-

#include <stdio.h>

int Numbers(int n);

int main() {

int num;

printf("Enter a positive integer: ");

scanf("%d", &num);

printf("Sum = %d", Numbers(num));

return 0;

}

int Numbers(int n) {

if (n != 0)

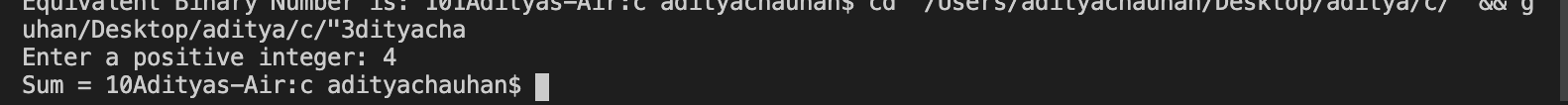
return n + Numbers(n - 1);

else

return n;

}

Output :-



practical:- 4

Aim :-wap calculate power to recursion.

Input:-

#include <stdio.h>

int power(int b, int a);

int main() {

int b, a, r;

printf("Enter base number: ");

scanf("%d", &b);

printf("Enter power number: ");

scanf("%d", &a);

r = power(b, a);

printf("%d^%d = %d", b, a, r);

return 0;

}

int power(int b, int a) {

if (a != 0)

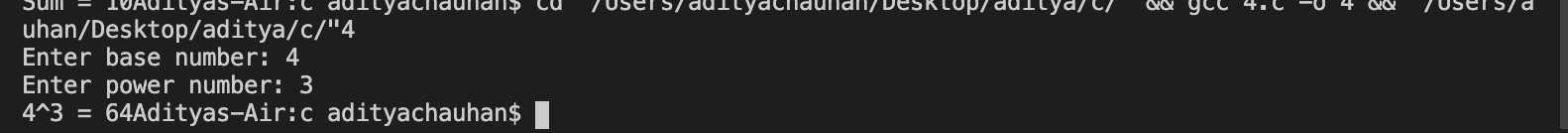
return (b \* power(b, a - 1));

else

return 1;

}

Output :-



practical:- 5

Aim :-Wap standard deviation

Input:-

#include <math.h>

#include <stdio.h>

float SD(float data[]);

int main() {

int i;

float data[10];

printf("Enter 10 elements: ");

for (i = 0; i < 10; ++i)

scanf("%f", &data[i]);

printf("\nStandard Deviation = %.6f", SD(data));

return 0;

}

float SD(float data[]) {

float sum = 0.0, mean, SD = 0.0;

int i;

for (i = 0; i < 10; ++i) {

sum += data[i];

}

mean = sum / 10;

for (i = 0; i < 10; ++i) {

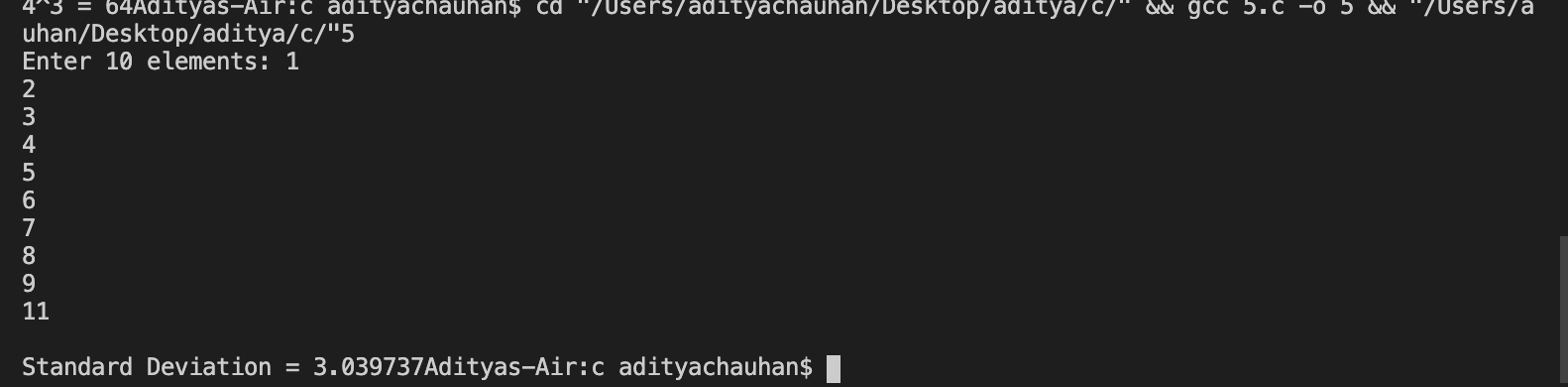
SD += pow(data[i] - mean, 2);

}

return sqrt(SD / 10);

}

Output :-



practical:- 6

Aim :-write c program to calculate average using arrays.

Input:-

#include<stdio.h>

int main() {

int n, i;

float num[100], sum = 0.0, avg;

printf("Enter the numbers of elements: ");

scanf("%d", &n);

while (n > 100 || n < 1) {

printf("Error! number should in range of (1 to 100).\n");

printf("Enter the number again: ");

scanf("%d", &n);

}

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

printf("%d. Enter number: ", i + 1);

scanf("%f", &num[i]);

sum += num[i];

}

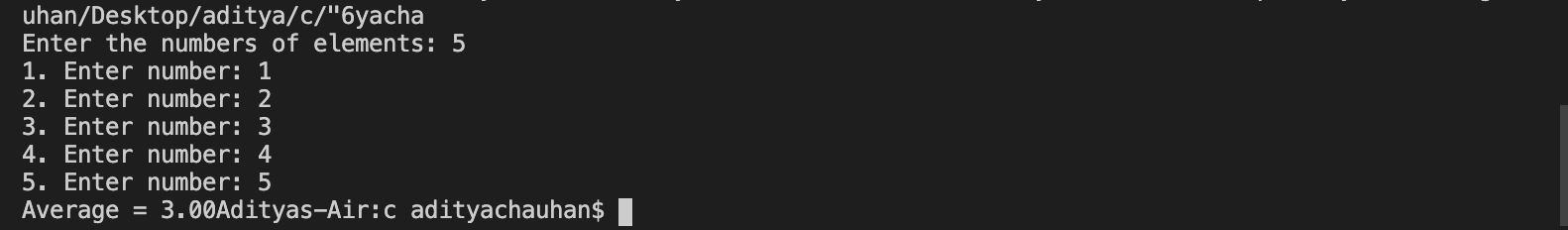
avg = sum / n;

printf("Average = %.2f", avg);

return 0;

}

Output :-



practical:- 7

Aim :-write a c program to accept and number from the user and sort them in dynamic memory allocation

Input:-

#include<stdio.h>

#include<stdlib.h>

int main()

{

int \*a,n,i,j,t;

printf("number sorted:");

scanf("%d",&n);

a=(int \*)malloc(n\*sizeof(int));

printf("enter %d number : \n",n);

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

{

scanf("%d",(a+i));

}

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

{

for ( j = 0; j < i; j++)

{

if (\*(a+i)<\*(a+j))

{

t=\*(a+i);

\*(a+i)=\*(a+j);

\*(a+j)=t;

}

}

}

printf("\n after sorting in ascending orde:\n");

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

{

printf("\n%d",\*(a+i));

}

return 0;

}

Output :-



practical:- 8

Aim :-write a program to add two matrix using multidimensional arrays

Input:-

#include <stdio.h>

int main() {

int r, c, a[100][100], b[100][100], sum[100][100], i, j;

printf("Enter the number of rows: ");

scanf("%d", &r);

printf("Enter the number of columns : ");

scanf("%d", &c);

printf("\nEnter elements of 1st matrix:\n");

for (i = 0; i < r; ++i)

for (j = 0; j < c; ++j) {

printf("Enter element a%d%d: ", i + 1, j + 1);

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

}

printf("Enter elements of 2nd matrix:\n");

for (i = 0; i < r; ++i)

for (j = 0; j < c; ++j) {

printf("Enter element b2%d%d: ", i + 1, j + 1);

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

}

// adding two matrices

for (i = 0; i < r; ++i)

for (j = 0; j < c; ++j) {

sum[i][j] = a[i][j] + b[i][j];

}

// printing the result

printf("\nSum of two matrices: \n");

for (i = 0; i < r; ++i)

for (j = 0; j < c; ++j) {

printf("%d ", sum[i][j]);

if (j == c - 1) {

printf("\n\n");

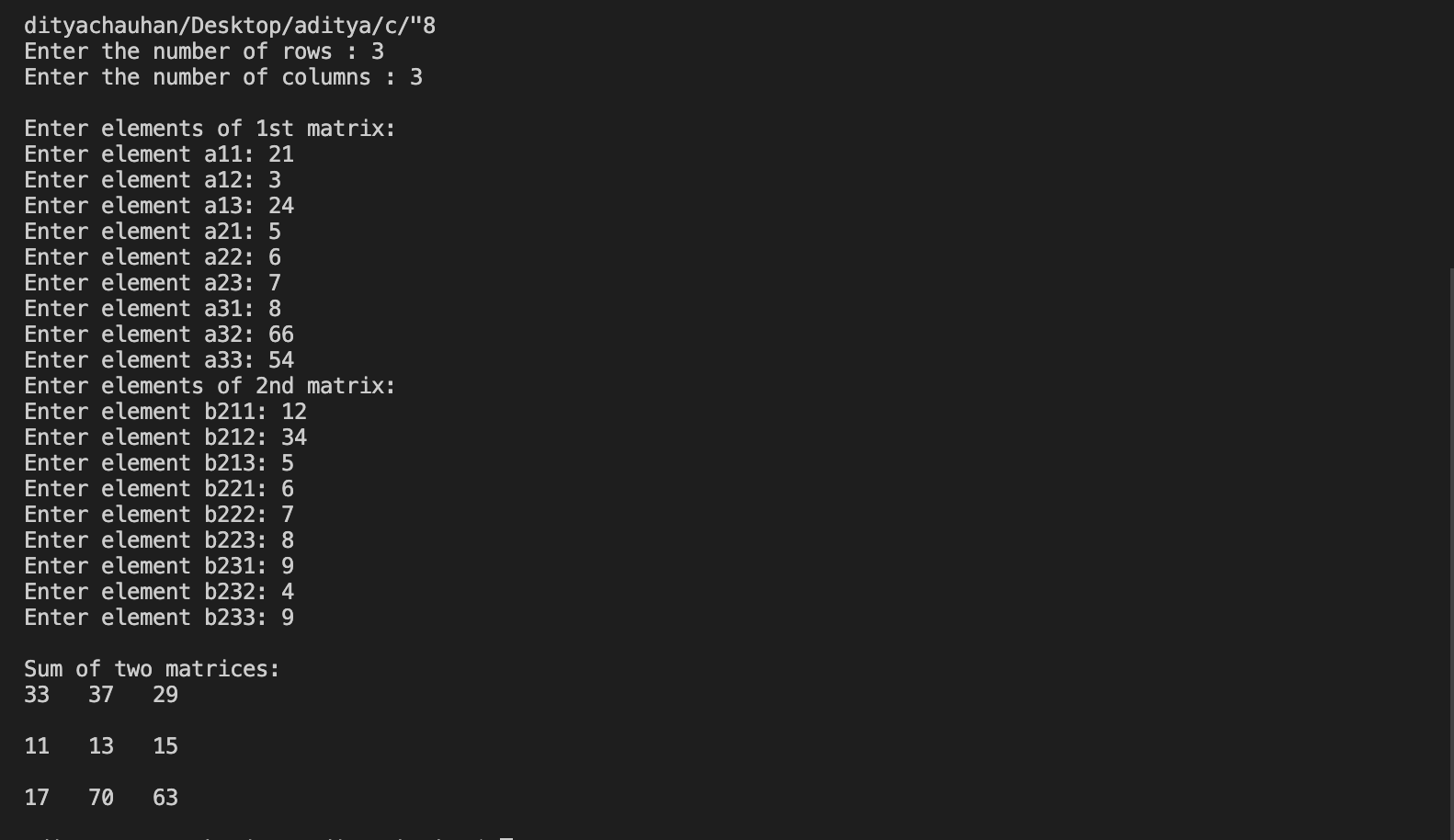
}

}

return 0;

}

Output :-



practical:- 9

Aim :-Wap multiplayer two matrix using multi dimensional array

Input:-

#include <stdio.h>

// function to get matrix elements entered by the user

void getMatrixElements(int matrix[][10], int row, int column) {

printf("\nEnter elements: \n");

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

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

printf("Enter a%d%d: ", i + 1, j + 1);

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

}

}

}

// function to multiply two matrices

void multiplyMatrices(int first[][10],

int second[][10],

int result[][10],

int r1, int c1, int r2, int c2) {

// Initializing elements of matrix mult to 0.

for (int i = 0; i < r1; ++i) {

for (int j = 0; j < c2; ++j) {

result[i][j] = 0;

}

}

// Multiplying first and second matrices and storing it in result

for (int i = 0; i < r1; ++i) {

for (int j = 0; j < c2; ++j) {

for (int k = 0; k < c1; ++k) {

result[i][j] += first[i][k] \* second[k][j];

}

}

}

}

// function to display the matrix

void display(int result[][10], int row, int column) {

printf("\nOutput Matrix:\n");

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

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

printf("%d ", result[i][j]);

if (j == column - 1)

printf("\n");

}

}

}

int main() {

int first[10][10], second[10][10], result[10][10], r1, c1, r2, c2;

printf("Enter rows and column for the first matrix: ");

scanf("%d %d", &r1, &c1);

printf("Enter rows and column for the second matrix: ");

scanf("%d %d", &r2, &c2);

// Taking input until

// 1st matrix columns is not equal to 2nd matrix row

while (c1 != r2) {

printf("Error! Enter rows and columns again.\n");

printf("Enter rows and columns for the first matrix: ");

scanf("%d%d", &r1, &c1);

printf("Enter rows and columns for the second matrix: ");

scanf("%d%d", &r2, &c2);

}

// get elements of the first matrix

getMatrixElements(first, r1, c1);

// get elements of the second matrix

getMatrixElements(second, r2, c2);

// multiply two matrices.

multiplyMatrices(first, second, result, r1, c1, r2, c2);

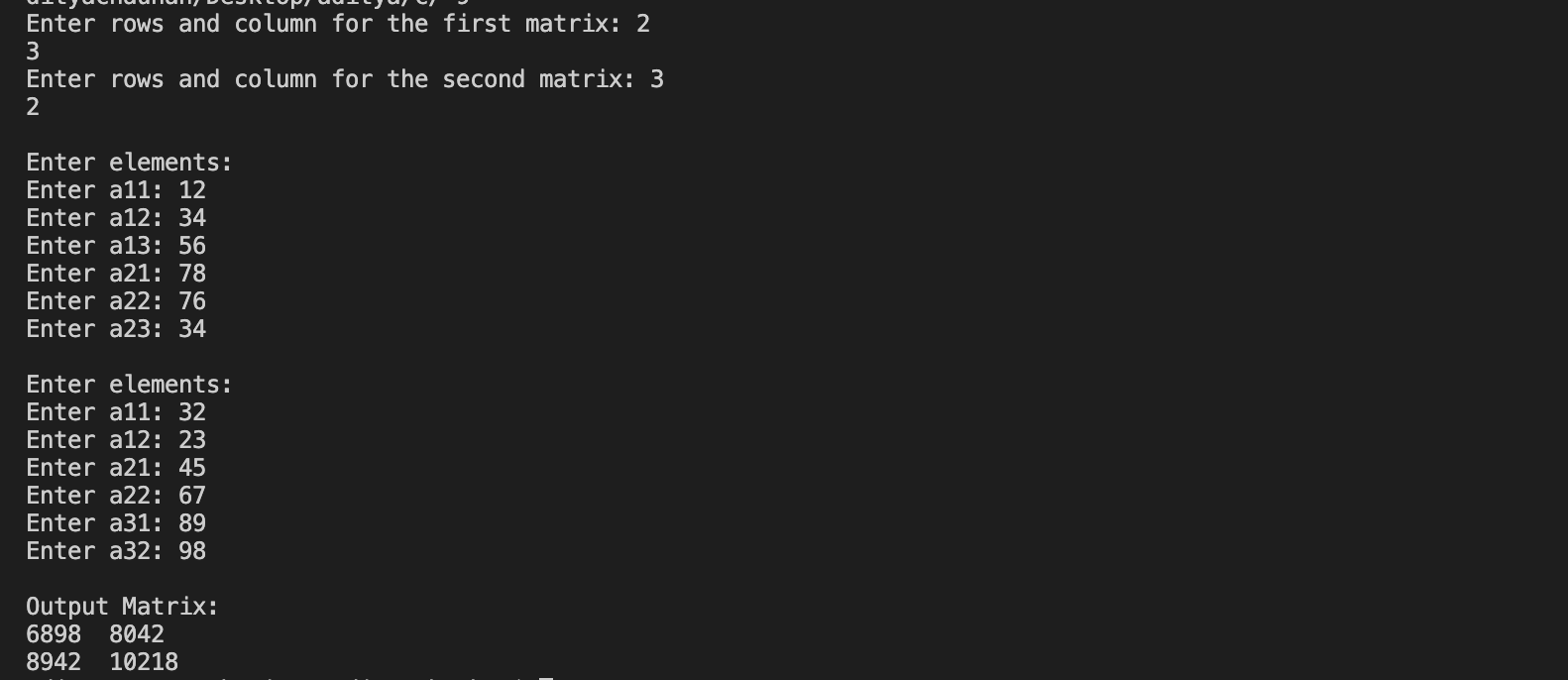
// display the result

display(result, r1, c2);

return 0;

}

Output :-



practical:- 10

Aim :-Wap transpose of a matrix

Input:-

#include <stdio.h>

int main() {

int a[100][100], transpose[100][100], r, c, i, j;

printf("Enter rows and columns: ");

scanf("%d %d", &r, &c);

// Assigning elements to the matrix

printf("\nEnter matrix elements:\n");

for (i = 0; i < r; ++i)

for (j = 0; j < c; ++j) {

printf("Enter element a%d%d: ", i + 1, j + 1);

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

}

// Displaying the matrix a[][]

printf("\nEntered matrix: \n");

for (i = 0; i < r; ++i)

for (j = 0; j < c; ++j) {

printf("%d ", a[i][j]);

if (j == c - 1)

printf("\n");

}

// Finding the transpose of matrix a

for (i = 0; i < r; ++i)

for (j = 0; j < c; ++j) {

transpose[j][i] = a[i][j];

}

// Displaying the transpose of matrix a

printf("\nTranspose of the matrix:\n");

for (i = 0; i < c; ++i)

for (j = 0; j < r; ++j) {

printf("%d ", transpose[i][j]);

if (j == r - 1)

printf("\n");

}

return 0;

}

Output :-

