**DSA LAB – 4**

**Name:** Etcherla Sai Manoj **Mis. No:** 112015044 **Branch:** CSE

**Question1:**

**Code:**

#include<iostream>

using namespace std;

void upperTri(int \*p, int rows, int columns){

cout << "----------------------------------------------------------\n";

// Matrix can have only triangular if it is a square matrix

if(rows == columns){

int flag;

// Checking elements which has to be zero

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

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

if(\*((p+i\*columns)+j) != 0) flag = 0;

else flag = 1;

}

}

if (flag == 1){

cout << "The matrix is a Upper Triangular Matrix" << endl;

}

else{

cout << "The matrix is not a Upper Triangular Matrix" << endl;

}

}

// Not a square matrix

else{

cout << "The matrix is a rectangular matrix. Triangular matrix does not exist." << endl;

}

cout << "----------------------------------------------------------\n\n";

}

void diasum(int \*p, int rows, int columns){

cout << "----------------------------------------------------------\n";

// Diagonal is defined for only square matrix

if(rows == columns){

int sum = 0;

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

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

// Diagonal elemants exist in same row and column

if(i == j){

sum = sum + \*((p+i\*columns)+j);

}

}

}

cout << "Sum Of Diagonal elements : " << sum << endl;;

}

// Not a square matrix

else{

cout << "The matrix is a rectangular matrix. Diagonal does not exist." << endl;

}

cout << "----------------------------------------------------------\n\n";

}

void transpose(int \*p, int rows, int columns){

cout << "----------------------------------------------------------\n";

// Define a transpose matrix with opposite order

int transpose[columns][rows];

// Reading transpose matrix

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

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

transpose[j][i] = \*((p+i\*columns)+j);

}

}

// Disaplay transpose matrix

cout << "Tranpose of the matrix is : " << endl;

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

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

cout << transpose[i][j] << " ";

}

cout << "\n";

}

cout << "----------------------------------------------------------\n\n";

}

void operations(int \*p, int rows, int columns){

// Define another matrix for addition, subtraction and multiplication operations

int rowsB, columnsB;

cout << "Enter number of rows and columns in matrix 2: ";

cin >> rowsB >> columnsB;

int matrix\_B[rowsB][columnsB];

// Reading another matrix

cout << "Enter elments of matrix 2" << endl;

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

cout << "Row " << i+1 << ": ";

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

cin >> matrix\_B[i][j];

}

}

cout << "\n----------------------------------------------------------\n";

// For Addition and subtraction, both matrices must have same order

if(rows == rowsB && columns == columnsB){

cout << "Addition of two matixes is :" << endl;

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

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

cout << \*((p+i\*columns)+j) + matrix\_B[i][j] << " ";

}

cout << "\n";

}

cout << "----------------------------------------------------------\n";

cout << "Subtraction of two matixes is :" << endl;

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

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

cout << \*((p+i\*columns)+j) - matrix\_B[i][j]<< " ";

}

cout << "\n";

}

}

// Don't have same order

else{

cout << "Addition and Subtraction is not possible" << endl;

}

cout << "----------------------------------------------------------\n\n";

// For mulitiplication, number of columns of first matrix should be equal to rows of second matrix

cout << "----------------------------------------------------------\n";

if(columns == rowsB){

int sum = 0;

cout << "Multiplication of two matrices is : " << endl;

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

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

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

sum = sum + \*((p+i\*columns)+k) \* matrix\_B[k][j];

}

cout << sum << " ";

sum = 0;

}

cout << "\n";

}

}

else{

cout << "Multiplication is not possible" << endl;

}

cout << "----------------------------------------------------------\n";

}

int main(){

int rows, columns;

cout << "Enter number of rows and columns in matrix : ";

cin >> rows >> columns;

int matrix[rows][columns];

// Taking input a matrix

cout << "Enter elments of matrix " << endl;

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

cout << "Row " << i+1 << ": ";

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

cin >> matrix[i][j];

}

}

upperTri((int \*)matrix, rows, columns);

diasum((int \*)matrix, rows, columns);

transpose((int \*)matrix, rows, columns);

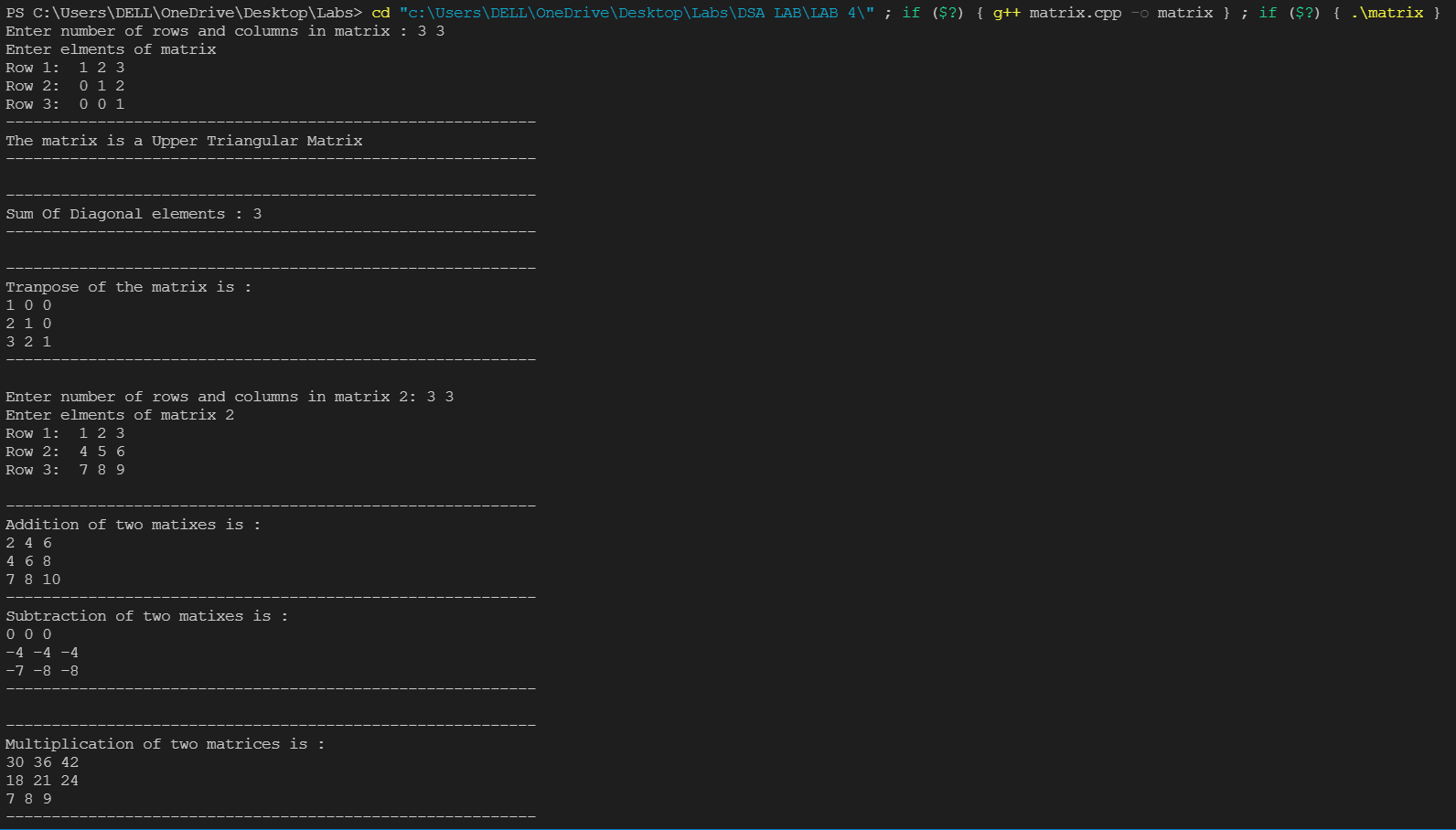
operations((int \*)matrix, rows, columns);

return 0;

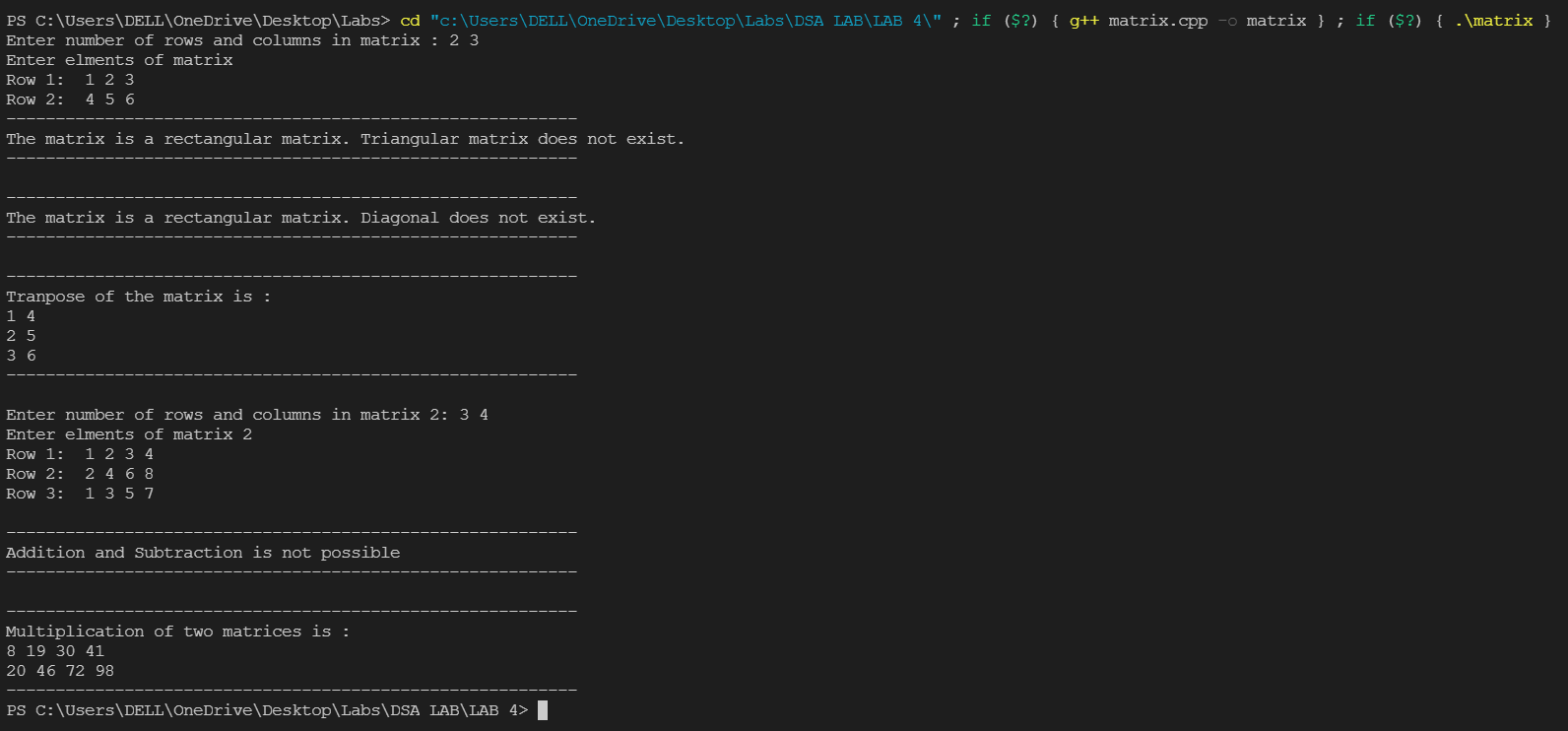
}

**Input & Output:**

Square matrix:

****

Rectangular matrix:



**Question 2:**

**Code:**

#include<iostream>

#include<stdlib.h>

using namespace std;

class poly {

public:

int \*coefficient, degree;

//function declaration

int getdata();

int display(int \*coefficient, int degree);

void addition(poly p1, poly p2);

void substraction(poly p1, poly p2);

void multiplication(poly p1, poly p2);

};

int poly::display(int \*coefficient, int degree) {

int i, j;

for (i = degree; i >= 0; i--) {

if(coefficient[i] >= 0) cout << coefficient[i] << "x^" << i;

else cout << "(" <<coefficient[i] << ")" << "x^" << i;

if ((i - 1) != -1)

cout << "+";

}

cout << "\n";

return 0;

}

int poly::getdata() {

int i;

cout << "Enter Degree Of Polynomial:";

cin >> degree;

coefficient = new int[degree + 1];

for (i = degree; i >= 0; i--) {

cout << "Enter coefficient of x^" << i << ":";

cin >> coefficient[i];

}

return 0;

}

void poly::addition(poly p1, poly p2) {

int max, i;

max = (p1.degree > p2.degree) ? p1.degree : p2.degree;

int \*sum = new int[max + 1];

if (p1.degree == p2.degree) {

for (i = p1.degree; i >= 0; i--)

sum[i] = p1.coefficient[i] + p2.coefficient[i];

}

if (p1.degree > p2.degree) {

for (i = p1.degree; i > p2.degree; i--)

sum[i] = p1.coefficient[i];

for (i = p2.degree; i >= 0; i--)

sum[i] = p1.coefficient[i] + p2.coefficient[i];

}

if (p1.degree < p2.degree) {

for (i = p2.degree; i > p1.degree; i--)

sum[i] = p2.coefficient[i];

for (i = p1.degree; i >= 0; i--)

sum[i] = p1.coefficient[i] + p2.coefficient[i];

}

cout << "\nAddition:";

display(sum, max);

cout << "\n";

}

void poly::substraction(poly p1, poly p2) {

int max, i;

max = (p1.degree > p2.degree) ? p1.degree : p2.degree;

int \*diff = new int[max + 1];

if (p1.degree == p2.degree) {

for (i = p1.degree; i >= 0; i--)

diff[i] = p1.coefficient[i] - p2.coefficient[i];

}

if (p1.degree > p2.degree) {

for (i = p1.degree; i > p2.degree; i--)

diff[i] = p1.coefficient[i];

for (i = p2.degree; i >= 0; i--)

diff[i] = p1.coefficient[i] - p2.coefficient[i];

}

if (p1.degree < p2.degree) {

for (i = p2.degree; i > p1.degree; i--)

diff[i] = -p2.coefficient[i];

for (i = p1.degree; i >= 0; i--)

diff[i] = p1.coefficient[i] - p2.coefficient[i];

}

cout << "\nSubstraction:";

display(diff, max);

cout << "\n";

}

void poly::multiplication(poly p1, poly p2) {

int i, j, max;

max = p1.degree + p2.degree;

int \*product = new int[max + 1]{0};

for (i = p1.degree; i >= 0; i--)

for (j = p2.degree; j >= 0; j--)

product[i + j] += p1.coefficient[i] \* p2.coefficient[j];

cout << "\nMultiplication:";

display(product, max);

}

int main() {

int choice;

poly p1, p2, p3;

cout << "Enter Polynomial 1:-" << endl;

p1.getdata();

cout << "Enter Polynomial 2:-" << endl;

p2.getdata();

while (1) {

cout << "\n\*\*\*\*\*\* Menu Selection \*\*\*\*\*\*" << endl;

cout << "1: Addition\n2: Substraction\n3: Multiplication\n0: Exit" << endl;

cout << "Enter your choice:";

cin >> choice;

switch (choice) {

case 1:

cout << "\n--------------- Addition ---------------\n";

cout << "Polynomial 1:";

p1.display(p1.coefficient, p1.degree);

cout << "Polynomial 2:";

p2.display(p2.coefficient, p2.degree);

p3.addition(p1, p2);

cout << "----------------------------------------\n";

break;

case 2:

cout << "\n------------- Substraction -------------\n";

cout << "Polynomial 1:";

p1.display(p1.coefficient, p1.degree);

cout << "Polynomial 2:";

p2.display(p2.coefficient, p2.degree);

p3.substraction(p1, p2);

cout << "----------------------------------------\n";

break;

case 3:

cout << "\n----------- Multiplication -------------\n";

cout << "Polynomial 1:";

p1.display(p1.coefficient, p1.degree);

cout << "Polynomial 2:";

p2.display(p2.coefficient, p2.degree);

p3.multiplication(p1, p2);

cout << "----------------------------------------\n";

break;

case 0:

return 0;

default:

cout << "\n----------- Enter a valid choice -------------\n";

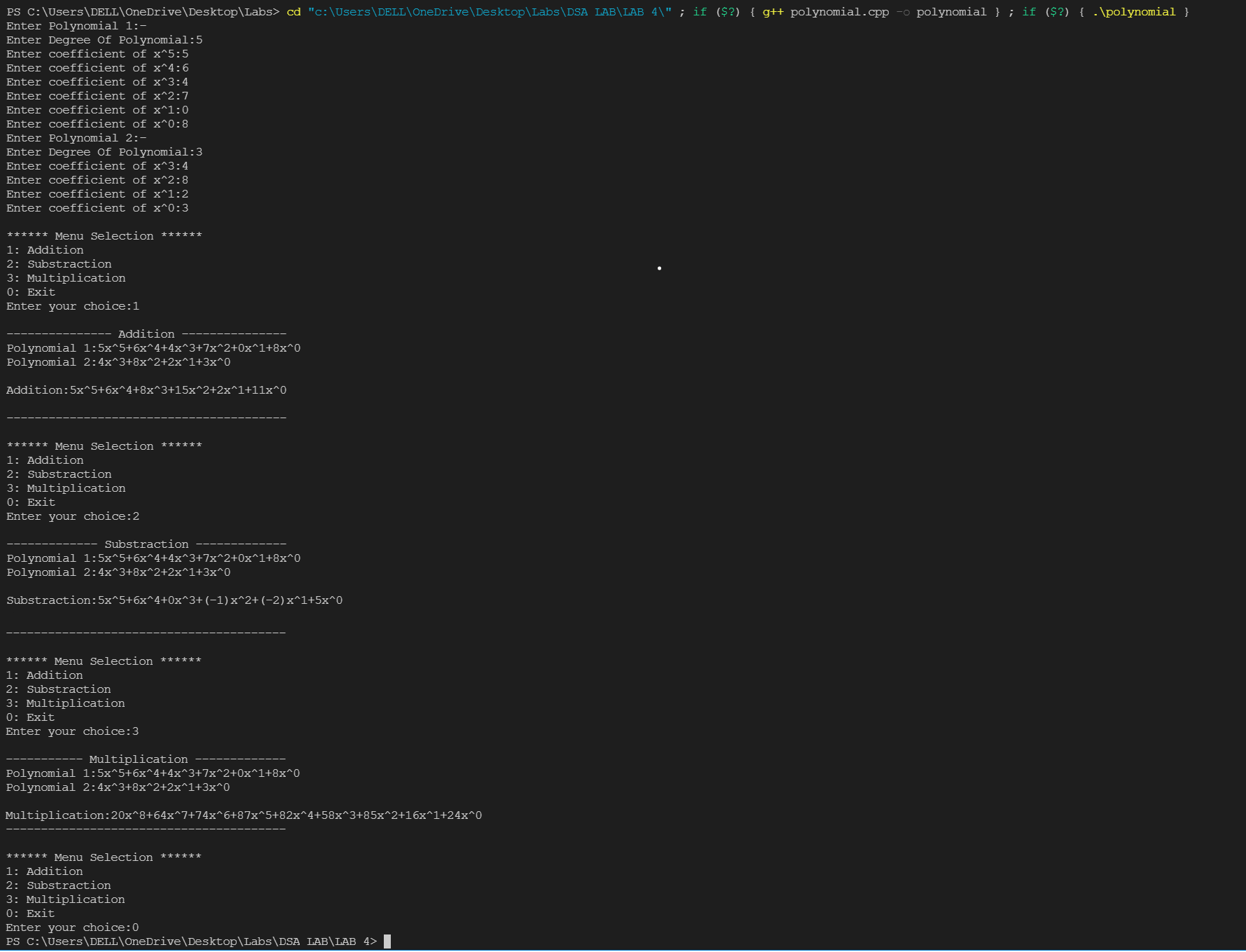
}

}

return 0;

}

**Input & Output:**

****