**Assignment No. 1**

**Problem Statement:**

Represent matrix using two-dimensional arrays and perform the following operations with

pointers: i. Addition ii. multiplication iii. transpose iv. Saddle point

**Objectives:** To write a neat code by following coding standards by selecting appropriate data structures

Outcomes: Students will be able to make use of coding standard for application development

**Code:**

#include<iostream>

using namespace std;

class Matrix{

int \*\*mat;

int row, col;

string name;

public:

Matrix(string name, int row, int col){

this->name = name;

this->row = row;

this->col = col;

mat = new int\*[row];

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

mat[i] = new int[col];

}

}

Matrix(string name){

this->name = name;

cout<<"Enter the dimensions for matrix "+this->name+" (row,col): ";

cin>>this->row>>this->col;

mat = new int\*[row];

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

mat[i] = new int[col];

}

}

void input(){

cout<<"Enter matrix- "+this->name+" :\n";

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

for(int j=0;j<col;j++)

cin>>this->mat[i][j];

}

void inputAgain(){

cout<<"Enter the dimensions for matrix "+this->name+" (row,col): ";

cin>>this->row>>this->col;

mat = new int\*[row];

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

mat[i] = new int[col];

}

input();

}

void print(){

cout<<"Matrix - "+this->name+" is:\n";

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

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

cout<<mat[i][j]<<"\t";

}

cout<<endl;

}

}

void set(int row, int col,int num){

mat[row][col] = num;

}

int get(int row, int col){

return mat[row][col];

}

int getRow(){

return this->row;

}

int getCol(){

return this->col;

}

void add(Matrix &m2){

if(this->col != m2.col || this->row != m2.row){

cout<<"Sorry, dimensions of the matrices do not match for addition !";

return;

}

cout<<"\nMatrix addition:\n";

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

for(int j=0;j<this->col;j++)

cout<<this->mat[i][j] + m2.mat[i][j]<<"\t";

cout<<endl;

}

return;

}

void sub(Matrix &m2){

if(this->col != m2.col || this->row != m2.row){

cout<<"Sorry, dimensions of the matrices do not match for subtraction !";

return;

}

cout<<"\nMatrix subtraction:\n";

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

for(int j=0;j<this->col;j++)

cout<<this->mat[i][j] - m2.mat[i][j]<<"\t";

cout<<endl;

}

return;

}

void mul(Matrix &m2){

if(this->col != m2.row){

cout<<"Sorry, dimensions of the matrices are not appropriate for multiplication !";

return;

}

Matrix m3("3",this->row,m2.col);

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

for(int j=0;j<m2.col;j++){

m3.mat[i][j]=0;

for(int k=0;k<this->col;k++){

m3.mat[i][j] = m3.mat[i][j] + this->mat[i][k] \* m2.mat[k][j];

}

}

}

cout<<"\nMatrix Multiplication:\n";

m3.print();

return;

}

void saddlePoint(){

int min, minIndex,max;

bool found = false;

cout<<"\nResult:\n";

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

min = this->mat[i][0];

minIndex = 0;

for(int j=0;j<this->col;j++){

if(this->mat[i][j]<min){

min = this->mat[i][j];

minIndex = j;

}

}

max = min;

for(int j=0;j<this->row;j++){

if(mat[j][minIndex]>max){

max = this->mat[j][minIndex];

}

}

if(max == min)

{

cout<<"\nFound saddle point: "<<max<<" at "<<i<<", "<<minIndex<<" position.";

found = true;

}

}

if(!found){

cout<<"\nNo saddle point found !";

}

return;

}

~Matrix(){

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

delete mat[i];

}

delete mat;

}

};

int main(){

Matrix m1("1"),m2("2");

int choice;

m1.input();

m2.input();

do{

cout<<"\n\n\n#### MENU ####";

cout<<"\n1. Input again \n2. Addition \n3. Subtraction \n4. Multiplication \n5. Saddle Point for First matrix \n6.Exit";

cout<<"\nChoice:";

cin>>choice;

switch(choice){

case 1:

m1.inputAgain();

m2.inputAgain();

break;

case 2:

m1.add(m2);

break;

case 3:

m1.sub(m2);

break;

case 4:

m1.mul(m2);

break;

case 5:

m1.saddlePoint();

break;

case 6:

cout<<"Shutting down...";

break;

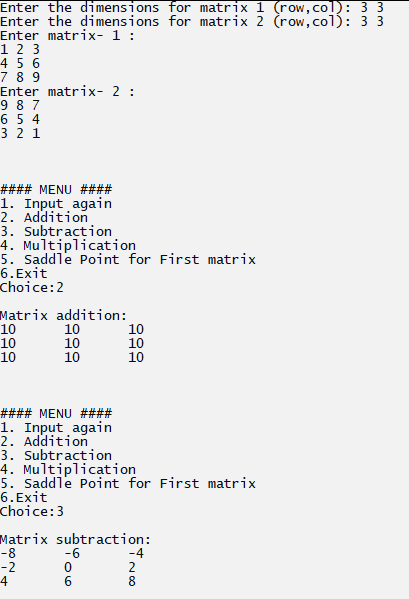
}

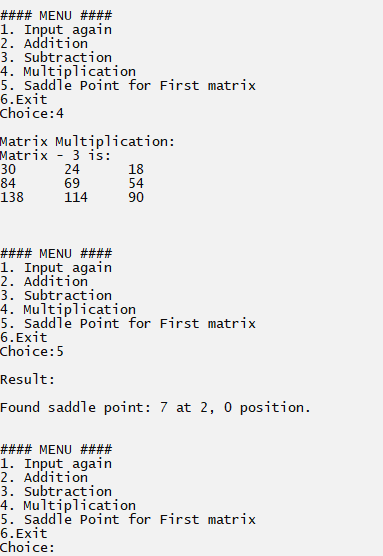
}while(choice!=6);

return 0;

**Output:**

1)

****

2)****