M. Ali. Arslan

19F-0348

**Major Assignment 1**

**Problem # 1**

#include<iostream>

#include<string>

using namespace std;

struct CourseGrade

{

string Name;

int Id;

int\* testScores;

float Average;

char Grade;

};

int main()

{

int Total\_Students, Total\_TestScores;

cout << "Enter number of Test Scores: ";

cin >> Total\_TestScores;

cout << "Enter number of Students: ";

cin >> Total\_Students;

CourseGrade\* students = new CourseGrade[Total\_Students];

for (int i = 0; i < Total\_Students; i++)

{

students[i].testScores = new int[Total\_TestScores];

}

cout << "\t\t\t\t\t\t\*Enter Data for each Student\*" << endl;

for (int i = 0; i < Total\_Students; i++)

{

cout << endl << endl;

cout << "Enter Name of student "<<i+1<<": ";

cin >> students[i].Name;

cout << "Enter Student ID: ";

cin >> students[i].Id;

cout << "\t\t\t\t\t\t \*Enter Test Scores\*" << endl;

for (int j = 0; j < Total\_TestScores; j++)

{

cout << "Enter Test Score # " << j + 1 << ":";

cin >> students[i].testScores[j];

}

}

int sum;

for (int i = 0; i < Total\_Students; i++)

{

sum = 0;

for (int j = 0; j < Total\_TestScores; j++)

{

sum = sum + students[i].testScores[j];

}

students[i].Average = static\_cast<float>(sum / Total\_TestScores);

}

for (int i = 0; i < Total\_Students; i++)

{

if (students[i].Average > 90)

{

students[i].Grade = 'A';

}

else if (students[i].Average > 80 && students[i].Average <= 90)

{

students[i].Grade = 'B';

}

else if (students[i].Average > 70 && students[i].Average <= 80)

{

students[i].Grade = 'C';

}

else if (students[i].Average > 60 && students[i].Average <= 70)

{

students[i].Grade = 'D';

}

else

{

students[i].Grade = 'F';

}

}

cout << " \t\t\t\t\t\tSTUDENTS DATA" << endl;

cout << "----------------------------------------------------------------------" << endl;

cout << "NAME\t\tID\t\t Average \t\tGrade" << endl;

cout << "----------------------------------------------------------------------" << endl;

for (int i = 0; i < Total\_Students; i++)

{

cout << students[i].Name << "\t\t" << students[i].Id << "\t\t " << students[i].Average << "\t\t\t "

<< students[i].Grade << endl;

}

}



**Problem # 2**

#include<iostream>

#include<string>

using namespace std;

struct student {

int id;

string Name;

float gpa;

int sem;

int dob;

student\* s;

string depart;

string gend;

};

void display(student\* d, int n);

void update(student\* w, int n1);

void byname(student\* e, int nu);

void duplicate(student\* f, int n);

void depart(student\* u, int n2);

void sem(student\* t, int n2);

void gend(student\* a, int n2);

void gpa(student\* v, int n2);

void old(student\* j, int nu1);

void resize(student l, int number);

int main()

{

student br;

int num = 2, i, c;

br.s = new student[num];

cout << "-------------------------WELCOME TO THE DATABASE OF THE STUDENTS -------------------" << endl;

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

{

cout << "ENTER THE ID OF THE STUDENT : ";

cin >> br.s[i].id;

cout << "ENTER THE NAME OF THE STUDENT : ";

cin >> br.s[i].Name;

cout << "ENTER THE DATE OF BIRTH OF THE STUDENT : ";

cin >> br.s[i].dob;

cout << "ENTER THE GENDER OF THE STUDENT : ";

cin >> br.s[i].gend;

cout << "ENTER THE SEMESTER OF THE STUDENT : ";

cin >> br.s[i].sem;

cout << "ENTER THE DEPARTMENT OF THE STUDENT : ";

cin >> br.s[i].depart;

cout << "ENTER THE GPA OF THE STUDENT : ";

cin >> br.s[i].gpa;

cout << "=============================================" << endl;

}

cout << "CHOOSE ANYONE OF YOUR OWN CHOICE : "

"\n 1-DISPLAY DATA OF ALL STUDENTS."

"\n 2-UPDATE DATA OF OLD STUDENT ."

"\n 3-ADD THE INFORMATION FOR NEW STUDENT."

"\n 4-SEARCH STUDENT BY ITS NAME . "

"\n 5-SEARCH STUDENT BY ANY THING ."

"\n 6-SEARCH AND DELETE INFORMATION OF OLD STUDENT."

"\n 7-SEARCH FOR DUPLICATED ENTRIES.";

cout << endl;

cout << "YOUR CHOICE IS : ";

cin >> c;

switch (c)

{

case 1:

{

cout << " ->THE INFORMATION OF THE STUDENTS IN THE TABULAR FOAM IS GIVEN AS :" << endl;

display(br.s, num);

cout << endl;

}break;

case 2:

{

cout << " -> SO, YOU WANT TO UPDATE THE INFORMATION OF THE OLD STUDENT !!" << endl;

update(br.s, num);

display(br.s, num);

cout << endl;

}break;

case 3:

{

cout << " ->>YOU WANTED TO ADD A NEW STUDENT " << endl;

resize(br, num);

cout << endl;

}break;

case 4: {

cout << " \*\*YOU WANT TO SEARCH THE STUDENT BY NAME : " << endl;

byname(br.s, num);

cout << endl;

}break;

case 5:

{int h;

cout << "YOU SEARCH THE STUDENTS ON PARTICULAR CHOICE" << endl;

cout << "SEARCH THE STUDENTS BY :"

"\n 1- PARTICULAR SEMESTER ."

"\n 2- PARTICULAR DEPARTMENT ."

"\n 3- PARTICULAR GENDER ."

"\n 4- PARTICULAR GPA ." << endl;

cout << "YOUR CHOICE IS : ";

cin >> h;

switch (h)

{

case 1:

{

sem(br.s, num);

cout << endl;

}break;

case 2: {

depart(br.s, num);

cout << endl;

}break;

case 3:

{

gend(br.s, num);

cout << endl;

}break;

case 4:

{

gpa(br.s, num);

cout << endl;

}

default:

{

cout << "YOU CHOOSE THE WRONG OPTION !!!" << endl;

}

}//h

cout << endl;

}break;

case 6:

{

cout << "YOU WANT TO DELETE THE INFORMATION OF AN OLD STUDENT!!" << endl;

old(br.s, num);

cout << endl;

}break;

case 7:

{cout << " ->FIND THE DUPLICATE ENTRY IN THE INFORMATION!" << endl;

duplicate(br.s, num);

cout << endl;

}break;

default:

{

cout << " ->>YOU CHOOSE THE WRONG OPTION!!!!" << endl;

}

}//s

cout << endl;

system("pause");

return 0;

}

void display(student\* d, int n)

{

cout << " ID" << " " << " NAME " << " " << "DOB " << " " << "GENDER " << " " << " SEMESTER" << " "

<< " DEPARTMENT " << " " << " GPA " << " " << endl;

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

{

cout << " " << d[i].id << " " << d[i].Name << " " << d[i].dob << " " << d[i].gend << " " << d[i].sem << " "

<< d[i].depart << " " << d[i].gpa << " " << endl;

}

}

void update(student\* w, int n1)

{

int r;

cout << "ENTER THE ID OF THE STUDENT OF WHICH YOU WANT TO UPDATE THE INFORMATION :";

cin >> r;

for (int q = 0; q < n1; q++)

{

if (r == w[q].id)

{

cout << " \*UPDATE THE DATE OF BIRTH OF THE STUDENT :";

cin >> w[q].dob;

cout << " \*UPDATE THE SEMESTER OF THE STUDENT : ";

cin >> w[q].sem;

cout << " \*UPDATE THE GPA OF THE STUDENT : ";

cin >> w[q].gpa;

cout << " \*UPDATE THE DEPARTMENT OF THE STUDENT : ";

cin >> w[q].depart;

}

}

cout << endl;

}

void byname(student\* e, int nu)

{

string str;

cout << "PLEASE ENTER THE NAME OF THE STUDENT OF WHICH YOU WANT TO SEARCH : ";

cin >> str;

for (int r = 0; r < nu; r++)

{

if (str == e[r].Name)

{

cout << " ->THE ID OF THE STUDENT IS : " << e[r].id << " " << endl;

cout << " ->THE DATE OF BIRTH OF THE STUDENT IS: " << e[r].dob << " " << endl;

cout << " ->THE SEMESTER OF THE STUDENT IS : " << e[r].sem << " " << endl;

cout << " ->THE GENDER OF THE STUDENT IS : " << e[r].gend << " " << endl;

cout << " ->THE GPA OF THE STUDENT IS : " << e[r].gpa << " " << endl;

cout << " ->THE DEPARTMENT OF THE STUDENT IS : " << e[r].depart << " " << endl;

}

}

cout << endl;

}

void duplicate(student\* f, int n)

{

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

{

if (f[i].id == f[i + 1].id)

{

delete[i] f;

cout << " ->THE FIRST ENTRY IS DELETED!!!" << endl;

}

}

cout << endl;

}

void sem(student\* t, int n2)

{

int p;

cout << "ENTER THE PARTICULAR SEMESTER : ";

cin >> p;

for (int i = 0; i < n2; i++)

{

if (p == t[i].sem)

{

cout << "THE NAME OF THE STUDENT IS : " << t[i].Name << " " << endl;

cout << "THE ID OF THE STUDENT IS : " << t[i].id << " " << endl;

cout << "THE DEPARTMENT OF THE STUDENT IS : " << t[i].depart << " " << endl;

cout << "THE GENDER OF THE STUDENT IS : " << t[i].gend << " " << endl;

cout << "THE GPA OF THE STUDENT IS : " << t[i].gpa << " " << endl;

}

cout << " \*\*" << endl;

}

}

void depart(student\* u, int n2)

{

string d;

cout << "ENTER THE PARTICULAR DEPARTMENT : ";

cin >> d;

for (int i = 0; i < n2; i++)

{

if (d == u[i].depart)

{

cout << "THE NAME OF THE STUDENT IS : " << u[i].Name << " " << endl;

cout << "THE ID OF THE STUDENT IS : " << u[i].id << " " << endl;

cout << "THE SEMESTER OF THE STUDENT IS : " << u[i].sem << " " << endl;

cout << "THE GENDER OF THE STUDENT IS : " << u[i].gend << " " << endl;

cout << "THE GPA OF THE STUDENT IS : " << u[i].gpa << " " << endl;

}

cout << " \*\*" << endl;

}

}

void gend(student\* a, int n2)

{

string d;

cout << "ENTER THE PARTICULAR gender : ";

cin >> d;

for (int i = 0; i < n2; i++)

{

if (d == a[i].gend)

{

cout << "THE NAME OF THE STUDENT IS : " << a[i].Name << " " << endl;

cout << "THE ID OF THE STUDENT IS : " << a[i].id << " " << endl;

cout << "THE SEMESTER OF THE STUDENT IS : " << a[i].sem << " " << endl;

cout << "THE DEPARTMENT OF THE STUDENT IS : " << a[i].depart << " " << endl;

cout << "THE GPA OF THE STUDENT IS : " << a[i].gpa << " " << endl;

}

cout << " \*\*" << endl;

}

}

void gpa(student\* v, int n2)

{

float r;

cout << "ENTER THE PARTICULAR GPA : ";

cin >> r;

for (int i = 0; i < n2; i++)

{

if (r == v[i].gpa)

{

cout << "THE NAME OF THE STUDENT IS : " << v[i].Name << " " << endl;

cout << "THE ID OF THE STUDENT IS : " << v[i].id << " " << endl;

cout << "THE SEMESTER OF THE STUDENT IS : " << v[i].sem << " " << endl;

cout << "THE DEPARTMENT OF THE STUDENT IS : " << v[i].depart << " " << endl;

cout << "THE gender OF THE STUDENT IS : " << v[i].gpa << " " << endl;

}

cout << " \*\*" << endl;

}

}

void old(student\* j, int nu1)

{

int m;

cout << "ENTER THE ID OF THE OLD STUDENT WHICH YOU WANT TO DELETE:";

cin >> m;

for (int o = 0; o < nu1; o++)

{

if (m == j[o].id)

{

delete[o]j;

}

}

}

void resize(student l, int number)

{

student\* ob = new student[number + 1];

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

{

ob[i].id = l.s[i].id;

ob[i].Name = l.s[i].Name;

ob[i].depart = l.s[i].depart;

ob[i].sem = l.s[i].sem;

ob[i].gpa = l.s[i].gpa;

ob[i].gend = l.s[i].gend;

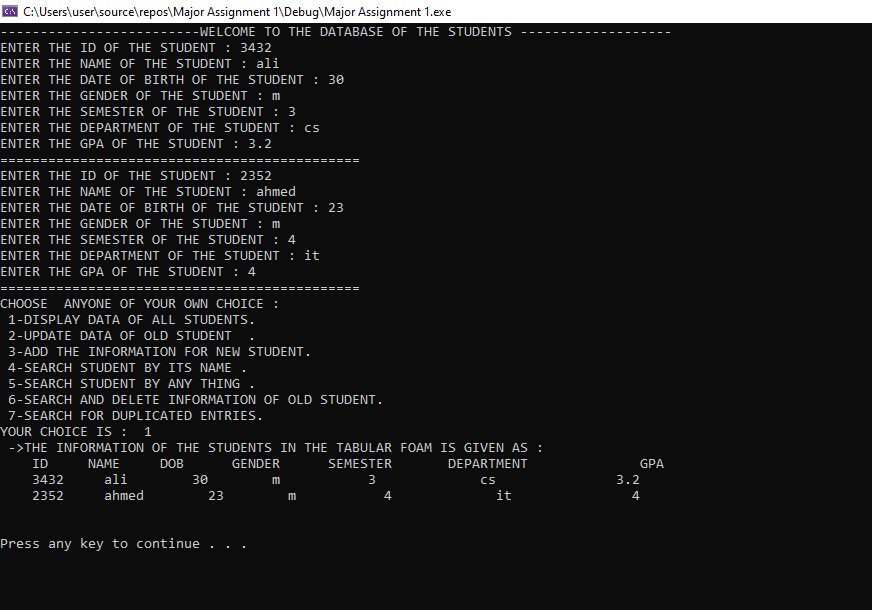
}

number++;

delete[] l.s;

l.s = ob;

}



**Problem # 3**

#include<iostream>

#include<cstring>

using namespace std;

void length(string in, string& min, string& max)

{

int len = in.length();

int n1 = 0, n2 = 0;

int minlen = len;

int index = 0;

int maxlen = 0;

int index1 = 0;

while (n2 <= len)

{

if (n2 < len && in[n2] != ' ')

{

n2++;

}

else

{

int word\_len = n2 - n1;

if (word\_len < minlen)

{

minlen = word\_len;

index = n1;

}

if (word\_len > maxlen)

{

maxlen = word\_len;

index1 = n1;

}

n2++;

n1 = n2;

}

}

min = in.substr(index, minlen);

max = in.substr(index1, maxlen);

}

int main()

{

string minWord, maxWord;

string a = " DSC is a Student Community";

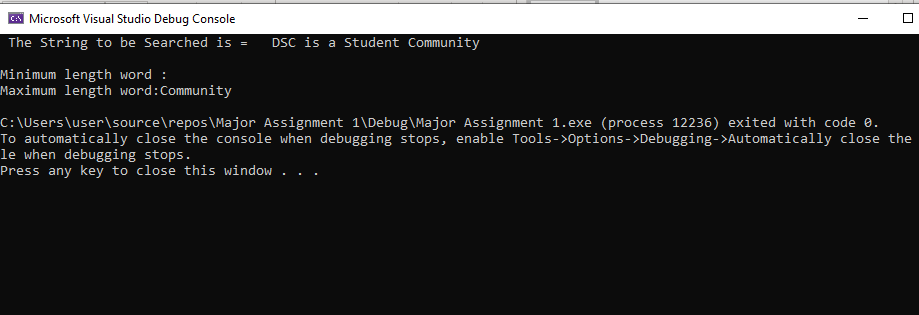
cout << " The String to be Searched is = " << " DSC is a Student Community" << endl << endl;

length(a, minWord, maxWord);

cout << "Minimum length word : " << minWord << endl;

cout << "Maximum length word:" << maxWord << endl;

}



**Problem # 4**

#include<iostream>

using namespace std;

const int MAX = 100;

void multiply(int row1, int col1, int A[][MAX], int row2, int col2, int b[][MAX], int c[][MAX]);

void output(int row1, int col1, int A[][MAX], int row2, int col2, int b[][MAX]);

int main()

{

int A[][MAX] = { {1, 2, 3},

{4, 5, 6},

{7, 8, 9} };

int b[][MAX] = { {1, 2, 3},

{4, 5, 6},

{7, 8, 9} };

int row1 = 3, col1 = 3, row2 = 3, col2 = 3;

cout << "Matrix A: " << endl;

for (int i = 0; i < row1; i++)

{

for (int j = 0; j < col1; j++)

{

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

}

cout << "\n";

}

cout << "\nMatrix B: " << endl;

for (int i = 0; i < row2; i++)

{

for (int j = 0; j < col2; j++)

{

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

}

cout << "\n";

}

cout << endl;

cout << "The Multiplication of Matrix A & B is: " << endl;

output(row1, col1, A, row2, col2, b);

}

void multiply(int row1, int col1, int A[][MAX], int row2, int col2, int b[][MAX], int c[][MAX]) //Recursively multiplying matrices

{

static int i = 0, j = 0, k = 0;

if (i >= row1)

{

return;

}

if (j < col2)

{

if (k < col1)

{

c[i][j] = c[i][j] + A[i][k] \* b[k][j];

k++;

multiply(row1, col1, A, row2, col2,

b, c);

}

k = 0;

j++;

multiply(row1, col1, A, row2, col2, b, c);

}

j = 0;

i++;

multiply(row1, col1, A, row2, col2, b, c);

}

void output(int row1, int col1, int A[][MAX], int row2, int col2, int b[][MAX]) //Display the output

{

if (row2 != col1)

{

cout << "Not Multipliable\n" << endl;

}

int c[MAX][MAX] = { 0 };

multiply(row1, col1, A, row2, col2, b, c);

for (int i = 0; i < row1; i++)

{

for (int j = 0; j < col2; j++)

{

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

}

cout << "\n";

}

}



**Problem # 5**

#include <iostream>

using namespace std;

void bubbleSort(int\*, int);

void Swap1(int\* arr, int\* i);

void print\_array(int\* data, int n);

void insertionSort(int\* arr, int len);

void print\_array1(int\* ptr, int size);

void swap(int\* ptr, int\* y);

void selectionSort(int\* ptr, int size);

int main()

{

int Array[5], size = 5, \* pointer;

cout << "Enter elements of an array: "<< endl;

for (int num = 0; num < size; num++)

{

cin >> Array[num];

}

pointer = Array;

cout << "\n\nThe array sorted by Bubble sorting is: " << endl;

bubbleSort(pointer, size);

cout << "\n\nThe array sorted by Bubble sorting is: " << endl;

selectionSort(pointer, size);

print\_array(pointer, size);

cout << "\n\nThe array sorted by Insertion sorting is: " << endl;

insertionSort(pointer, size);

print\_array1(pointer, size);

}

void bubbleSort(int\* pointer, int size) //Bubble Sorting

{

int count1, count2, swap;

for (count1 = 0; count1 < size - 1; count1++)

{

for (count2 = 0; count2 < size - count1 - 1; count2++)

{

if (\*(pointer + count2) > \* (pointer + count2 + 1))

{

swap = \*(pointer + count2);

\*(pointer + count2) = \*(pointer + count2 + 1);

\*(pointer + count2 + 1) = swap;

}

}

}

for (count1 = 0; count1 < size; count1++)

{

cout << \*(pointer + count1) << " , ";

}

}

void selectionSort(int\* ptr, int size) //Selection Sorting

{

int smallest;

for (int i = 0; i < size - 1; i++)

{

smallest = i;

for (int index = i + 1; index < size; index++)

{

if (ptr[index] < ptr[smallest])

{

smallest = index;

}

}

swap(ptr[i], ptr[smallest]);

}

}

void swap(int\* ptr, int\* y) //Swapping for selection sorting

{

int temp = \*ptr;

\*ptr = \*y;

\*y = temp;

}

void print\_array(int\* ptr, int size) //Printing selection sorted array

{

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

{

cout << \*ptr << " , ";

ptr++;

}

}

void insertionSort(int\* ptr, int size) //Insertion Sorting

{

int\* last = ptr + size;

for (int\* i = ptr + 1; i < last; i++)

{

if (\*i < \*(i - 1))

{

Swap1(ptr, i);

}

}

}

void Swap1(int\* ptr, int\* i) //Swapping for insertion sorting

{

int j;

for (j = \*i; i > ptr && \*(i - 1) > j; i--)

{

\*i = \*(i - 1);

}

\*i = j;

}

void print\_array1(int\* ptr, int size) //Printing insertion sorted array

{

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

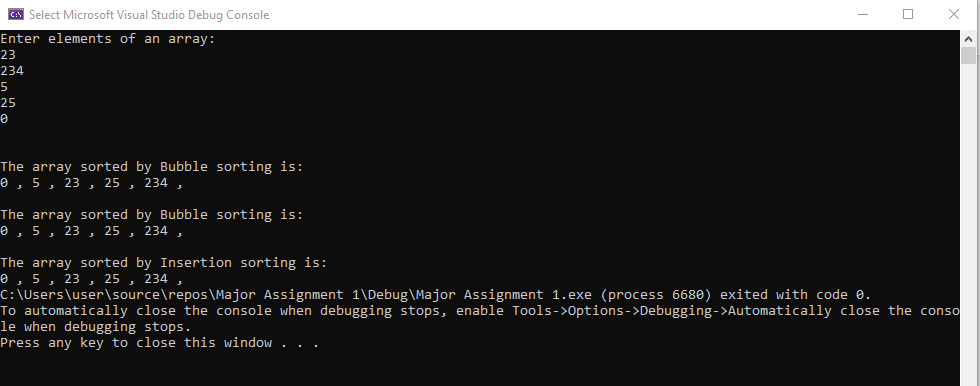
{

cout << \*ptr << " , ";

ptr++;

}

}



**Problem # 6**

#include <iostream>

#include <iomanip>

#include <ctime>

#include <cstdlib>

using namespace std;

void add(int\* mat1, int\* mat2, int row1, int col1);

void subs(int\* mat1, int\* mat2, int row1, int col1);

void mult(int\* mat1, int\* mat2, int row1, int col2);

void transpose(int\* mat1, int\* mat2, int row1, int col1, int row2, int col2);

int main()

{

srand(time(0));

int row1, col1, row2, col2;

cout << "Enter the rows of first matrix: ";

cin >> row1;

cout << "Enter the columns of first matrix: ";

cin >> col1;

cout << "Enter the rows of second matrix: ";

cin >> row2;

cout << "Enter the columns of second matrix: ";

cin >> col2;

int\* mat1 = new int[row1 \* col1];

int\* mat2 = new int[row2 \* col2];

//Initializing the matrices with random values

for (int i = 0; i < row1; i++)

{

for (int j = 0; j < col1; j++)

{

\*(mat1 + i \* col1 + j) = rand() % 10 + 1;

}

}

cout << "First Matrix" << endl;

for (int i = 0; i < row1; i++)

{

for (int j = 0; j < col1; j++)

{

cout << setw(3) << \*(mat1 + i \* col1 + j) << " ";

}

cout << endl;

}

for (int i = 0; i < row2; i++)

{

for (int j = 0; j < col2; j++)

{

\*(mat2 + i \* col2 + j) = rand() % 10 + 1;

}

cout << endl;

}

cout << "Second Matrix" << endl;

for (int i = 0; i < row2; i++)

{

for (int j = 0; j < col2; j++)

{

cout << setw(3) << \*(mat2 + i \* col2 + j) << " ";

}

cout << endl;

}

int n;

cout << "Choose the option!\n";

cout << "1) Addition\n2) Substraction\n3) Multiplication\n4) Transpose\n\n";

cin >> n;

switch (n)

{

case 1:

if (row1 == row2 && col1 == col2)

add(mat1, mat2, row1, col1);

break;

case 2:

if (row1 == row2 && col1 == col2)

subs(mat1, mat2, row1, col1);

break;

case 3:

if (col1 == row2)

mult(mat1, mat2, row1, col2);

else

cout << "Number of columns of first matrix is not equal to number of rows of second\n";

break;

case 4:

transpose(mat1, mat2, row1, row2, col1, col2);

break;

default:

break;

}

// deallocate memory

delete[] mat1;

delete[] mat2;

}

void add(int\* mat1, int\* mat2, int row1, int col1)

{

int\* mat3 = new int[row1 \* col1];

for (int i = 0; i < row1; i++)

{

for (int j = 0; j < col1; j++)

{

\*(mat3 + i \* col1 + j) = \*(mat1 + i \* col1 + j) + \*(mat2 + i \* col1 + j);

}

}

cout << "Addition Matrix " << endl;

for (int i = 0; i < row1; i++)

{

for (int j = 0; j < col1; j++)

{

cout << setw(3) << \*(mat3 + i \* col1 + j) << " ";

}

cout << endl;

}

delete[] mat3;

}

void subs(int\* mat1, int\* mat2, int row1, int col1)

{

int\* mat3 = new int[row1 \* col1];

for (int i = 0; i < row1; i++)

{

for (int j = 0; j < col1; j++)

{

\*(mat3 + i \* col1 + j) = \*(mat1 + i \* col1 + j) - \*(mat2 + i \* col1 + j);

}

}

cout << "Substraction Matrix " << endl;

for (int i = 0; i < row1; i++)

{

for (int j = 0; j < col1; j++)

{

cout << setw(3) << \*(mat3 + i \* col1 + j) << " ";

}

cout << endl;

}

delete[] mat3;

}

void mult(int\* mat1, int\* mat2, int row1, int col2)

{

int\* mat3 = new int[row1 \* col2];

for (int i = 0; i < row1; i++)

{

for (int j = 0; j < col2; j++)

{

\*(mat3 + i \* col2 + j) = \*(mat1 + i \* row1 + j) \* \*(mat2 + i \* col2 + j);

}

}

cout << "Multiplied Matrix: \n";

for (int i = 0; i < row1; i++)

{

for (int j = 0; j < col2; j++)

{

cout << setw(3) << \*(mat3 + i \* col2 + j) << " ";

}

cout << endl;

}

delete[] mat3;

}

void transpose(int\* mat1, int\* mat2, int row1, int col1, int row2, int col2)

{

int\* mat3 = new int[col1 \* row1];

for (int i = 0; i < col1; i++)

{

for (int j = 0; j < row1; j++)

{

\*(mat3 + i \* col1 + j) = \*(mat1 + i \* row1 + j);

}

cout << endl;

}

cout << "Transpose of first Matrix: \n";

for (int i = 0; i < col1; i++)

{

for (int j = 0; j < row1; j++)

{

cout << setw(3) << \*(mat3 + j \* row1 + i) << " ";

}

cout << endl;

}

int\* mat4 = new int[col2 \* row2];

for (int i = 0; i < col2; i++)

{

for (int j = 0; j < row2; j++)

{

\*(mat4 + i \* col2 + j) = \*(mat2 + i \* row2 + j);

}

cout << endl;

}

cout << "Transpose of second Matrix: \n";

for (int i = 0; i < col2; i++)

{

for (int j = 0; j < row2; j++)

{

cout << setw(3) << \*(mat4 + j \* row2 + i) << " ";

}

cout << endl;

}

delete[] mat4;

delete[] mat3;

}

