**DS LAB 2**

QUESTION 1:

#include<iostream>

using namespace std;

int main()

{

    //making initially an array

    int\*\* jaggedArr = new int\* [5];

    int\* sizes = new int[5] {5, 5, 5, 5, 5};

    for (int i = 0; i < 5; i++)

    {

        jaggedArr[i] = new int[sizes[i]];

        for (int j = 0; j < sizes[i]; j++)

        {

            jaggedArr[i][j] = i + j; //random initialization

        }

    }

    //Before resizing

    for (int i = 0; i < 5; i++)

    {

        for (int j = 0; j < 5; j++)

        {

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

        }

        cout << endl;

    }

    for (int i = 0; i < 5; i++)

    {

        int\* temp = new int[10];

        for (int j = 0; j < sizes[i]; j++)

        {

            temp[j] = jaggedArr[i][j];

        }

        delete[] jaggedArr[i];

        jaggedArr[i] = temp;

        sizes[i] = 10;

    }

    for (int i = 0; i < 5; i++)

    {

        for (int j = 5; j < sizes[i]; j++)

        {

            jaggedArr[i][j] = i + j;

        }

    }

    //After resizing

    for (int i = 0; i < 5; i++)

    {

        for (int j = 0; j < 10; j++)

        {

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

        }

        cout << endl;

    }

    for (int i = 0; i < 5; i++)

    {

        delete[] jaggedArr[i];

    }

    delete[] jaggedArr;

}

QUESTION 2:

Header File:

#ifndef MATRIX\_MULTIPLY\_H

#define MATRIX\_MULTIPLY\_H

namespace N

{

    class matrix\_multiply

    {

    public:

        int\*\* MultiplyMatrix( int\*\* matrix1, int\*\* matrix2, int row1, int row2, int col1, int col2 );

        void InputMatrix( int\*\* m, int row, int col );

        void PrintMatrix( int\*\* resultant, int row, int col );

    };

}

#endif

Cpp File:

#include "matrix\_multiply.h"

#include<iostream>

using namespace std;

using namespace N;

int\*\* matrix\_multiply::MultiplyMatrix( int\*\* matrix1, int\*\* matrix2, int row1, int row2, int col1, int col2 )

{

    if (col1 == row2)

    {

        int\*\* multipliedMatrix = new int\* [row1];

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

        {

            multipliedMatrix[i] = new int[col2];

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

            {

                multipliedMatrix[i][j] = 0;

            }

        }

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

        {

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

            {

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

                {

                    multipliedMatrix[i][j] += matrix1[i][k] \* matrix2[k][j];

                }

            }

        }

        return multipliedMatrix;

    }

    else

    {

        cout<<"Matrices can't be multiplied"<<endl;

        return nullptr;

    }

}

void matrix\_multiply::InputMatrix( int\*\* m, int row, int col )

{

    cout<<"----MATRIX INPUT-----"<<endl;

    for (int i = 0; i < row; i++)

    {

        cout << "Enter vals for row " << i + 1 << ": " << endl;

        for (int j = 0; j < col; j++)

        {

            cin >> m[i][j];

        }

    }

}

void matrix\_multiply::PrintMatrix( int\*\* resultant, int row, int col )

{

    cout << "Resultant multiplied matrix: " << endl;

    for (int i = 0; i < row; i++)

    {

        for (int j = 0; j < col; j++)

        {

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

        }

        cout << endl;

    }

}

void AllocateMatrix(int\*\* &matrix, int row, int col)

{

    matrix = new int\* [row];

    for (int i = 0; i < row; i++)

    {

        matrix[i] = new int[col];

    }

}

void DeallocateMatrix( int\*\* matrix, int row ) {

    for (int i = 0; i < row; i++) {

        delete[] matrix[i];

    }

    delete[] matrix;

}

int main()

{

    matrix\_multiply mm;

    int row1 = 3;

    int col1 = 3;

    int\*\*m1;

    AllocateMatrix(m1, row1, col1);

    int row2 = 3;

    int col2 = 3;

    int\*\* m2;

    AllocateMatrix( m2, row2, col2 );

    mm.InputMatrix(m1, row1, col1);

    mm.InputMatrix(m2, row2, col2);

    int\*\* multipliedMatrix = mm.MultiplyMatrix( m1, m2, row1, row2, col1, col2);

    mm.PrintMatrix(multipliedMatrix, row1, col2);

    DeallocateMatrix( m1, row1 );

    DeallocateMatrix( m2, row2 );

    DeallocateMatrix( multipliedMatrix, row1 );

}

QUESTION 3:

#include<iostream>

using namespace std;

bool AreFriends( int people1, int people2, int sizeOfArr, bool arr[][5] )

{

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

    {

        if (arr[people1][i] == true && arr[people2][i] == true)

        {

            return true;

        }

    }

    return false;

}

int main()

{

    // Initializing the friendship array as per the problem statement

    bool arr2d[5][5] = {

        {true, true, false, true, true},

        {true, true, true, false, false},

        {false, true, true, false, false},

        {true, false, false, true, true},

        {true, false, false, true, true}

    };

    bool ans = AreFriends( 0, 4, 5, arr2d );

    cout << "Do person 0 and person 4 have a common friend? " << (ans ? "Yes" : "No") << endl;

}

QUESTION 4:

#include<iostream>

using namespace std;

int main()

{

    //IMPLEMENTING JAGGED ARRAY

    float\*\* gpa = new float\* [4];

    int courses[4] = { 3, 4, 2, 1 };

    int deptSize = 4;

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

    {

        gpa[i] = new float[courses[i]];

    }

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

    {

        cout << "Enter gpa of each course for dept "<<i+1<<": " << endl;

        for (int j = 0; j < courses[i]; j++)

        {

            cin >> gpa[i][j];

        }

    }

    for (int i = 0; i < 4; i++)

    {

        switch (i)

        {

        case 0:

            cout << "SE dept: " << endl;

            break;

        case 1:

            cout << "AI deppt: " << endl;

            break;

        case 2:

            cout << "CS dept: " << endl;

        case 3:

            cout << "DS dept: " << endl;

            break;

        }

        for (int j = 0; j < courses[i]; j++)

        {

            cout << gpa[i][j] << " ";

        }

        cout << endl;

    }

    for (int i = 0; i < 4; i++)

    {

        delete[] gpa[i];

    }

    delete[] gpa;

}

QUESTION 5:

#include<iostream>

using namespace std;

int main()

{

    string\*\* attendees = new string \* [4];

    int seatingCapacity[4] = { 3, 4, 2, 1 };

    for (int i = 0; i < 4; i++)

    {

        attendees[i] = new string[seatingCapacity[i]];

    }

    for (int i = 0; i < 4; i++)

    {

        cout << "Enter vals for dept "<<i+1<<": " << endl;

        for (int j = 0; j < seatingCapacity[i]; j++)

        {

            cin >> attendees[i][j];

        }

    }

    for (int i = 0; i < 4; i++)

    {

        cout << "Names of attandees' sitting in " << i + 1 << " row: " << endl;

        for (int j = 0; j < seatingCapacity[i]; j++)

        {

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

        }

        cout << endl;

    }

    for (int i = 0; i < 4; i++)

    {

        delete[] attendees[i];

    }

    delete[] attendees;

}