Muhammad Mufeez
23k-0800
BCS-3J
DS Lab 2 Tasks

Task 0:

```
#include <bits/stdc++.h>

using namespace std;

class DynamicMatrix
{
public:
    int rows, cols;
    int **bptr;
    DynamicMatrix(int rows, int cols)
    {
        this->rows = rows;
        this->cols = cols;
        bptr = new int *[rows];

        for (int i = 0; i < rows; i++)
        {
            bptr[i] = new int[cols];
        }

        for (int i = 0; i < rows; ++i)
        {
            for (int j = 0; j < cols; ++j)
            {
                *(*(bptr + i) + j) = i + j;
            }
        }
        printMatrix();
    }
    void resizeMatrix(int newRows, int newCols)
    {
        int **temp = new int *[newRows];
        for (int i = 0; i < newRows; i++)
        {
```

```
            temp[i] = new int[newCols];
        }
        for (int i = 0; i < min(rows, newRows); i++)
        {
            for (int j = 0; j < min(cols, newCols); j++)
            {
                temp[i][j] = bptr[i][j];
            }
        }
        for (int i = 0; i < rows; ++i)
        {
            delete[] bptr[i];
        }
        delete[] bptr;
        bptr = temp;
        // updating new cells values
        for (int i = 0; i < newRows; ++i)
        {
            for (int j = 0; j < newCols; ++j)
            {
                if (j >= cols)
                {
                    bptr[i][j] = i + j;
                }
                if (i >= rows)
                {
                    bptr[i][j] = i + j;
                }
            }
        }
        this->rows = newRows;
        this->cols = newCols;
        printMatrix();
    }
    void printMatrix()
    {
        // printing without increment
        cout << "printing without increment" << endl;
        for (int i = 0; i < rows; ++i)
        {
```

```
            for (int j = 0; j < cols; ++j)
            {
                cout << *(*(bptr + i) + j) << " ";
            }
            cout << endl;
        }
        cout << endl
             << endl;
        // incremented at odd indices
        cout << "incremented at odd indices" << endl;
        for (int i = 0; i < rows; ++i)
        {
            for (int j = 0; j < cols; ++j)
            {
                if (j % 2 == 1)
                    cout << (*(*(bptr + i) + j) + 2) << " ";
                else
                    cout << *(*(bptr + i) + j) << " ";
            }
            cout << endl;
        }
    }
    void TransposeMatrix()
    {
        // for a transpose matrix, rows and cols should be equal
        int minRows = min(rows, cols);
        int **temp = new int *[minRows];
        for (int i = 0; i < rows; i++)
        {
            temp[i] = new int(minRows);
        }
        cout << "Transposed matrix: " << endl;
        for (int i = 0; i < minRows; i++)
        {
            for (int j = 0; j < minRows; ++j)
            {
                temp[i][j] = bptr[j][i];
                cout << temp[i][j] << " ";
            }
            cout << endl;
```

```
        }
        for (int i = 0; i < minRows; ++i)
        {
            delete[] temp[i];
        }
        delete[] temp;
    }
    ~DynamicMatrix()
    {
        for (int i = 0; i < rows; ++i)
        {
            delete[] bptr[i];
        }
        delete[] bptr;
    }
};
int main()
{
    DynamicMatrix *obj1 = new DynamicMatrix(3, 4);
    cout << "calling resize matrix function(small size)" << endl;
    obj1->resizeMatrix(2, 2);
    cout << "calling resize matrix function(greater size)" << endl;
    obj1->resizeMatrix(3, 5);
    cout << "Transposing the matrix" << endl;
    obj1->TransposeMatrix();
    return 0;
}
```

```
2 5 4 7
calling resize matrix function(small size)
printing without increment
0 1
1 2


incremented at odd indices
0 3
1 4
calling resize matrix function(greater size)
printing without increment
0 1 2 3 4
1 2 3 4 5
2 3 4 5 6


incremented at odd indices
0 3 2 5 4
1 4 3 6 5
2 5 4 7 6
Transposing the matrix
Transposed matrix:
0 1 2
1 2 3
2 3 4

--------------------------------
Process exited after 0.2806 seconds with return value 0
Press any key to continue . . .
```

Task 1:

```cpp
#include <bits/stdc++.h>



using namespace std;

class DynamicMatrix
{
public:
    int rows, cols;
    int **bptr;
    DynamicMatrix(int rows, int cols)
    {
        this->rows = rows;
        this->cols = cols;
        bptr = new int *[rows];
```

```
    for (int i = 0; i < rows; i++)
    {
        bptr[i] = new int[cols];
    }

    for (int i = 0; i < rows; ++i)
    {
        for (int j = 0; j < cols; ++j)
        {
            *(*(bptr + i) + j) = i + j;
        }
    }
    printMatrix();
}
void resizeMatrix(int newRows, int newCols)
{
    int **temp = new int *[newRows];
    for (int i = 0; i < newRows; i++)
    {
        temp[i] = new int[newCols];
    }
    for (int i = 0; i < min(rows, newRows); i++)
    {
        for (int j = 0; j < min(cols, newCols); j++)
        {
            temp[i][j] = bptr[i][j];
        }
    }
    for (int i = 0; i < rows; ++i)
    {
        delete[] bptr[i];
    }
    delete[] bptr;
    bptr = temp;
        for (int i = 0; i < newRows; ++i)
        {
            for (int j = 0; j < newCols; ++j)
            {
                if (j >= cols)
```

```
                    {
                        bptr[i][j] = i + j;
                    }
                    if (i >= rows)
                    {
                        bptr[i][j] = i + j;
                    }
                }
            }
        this->rows = newRows;
        this->cols = newCols;
        printMatrix();
    }
    void printMatrix()
    {
        // printing without increment
        cout << "printing without increment" << endl;
        for (int i = 0; i < rows; ++i)
        {
            for (int j = 0; j < cols; ++j)
            {
                cout << *(*(bptr + i) + j) << " ";
            }
            cout << endl;
        }
        cout << endl;
    }
    ~DynamicMatrix()
    {
        for (int i = 0; i < rows; ++i)
        {
            delete[] bptr[i];
        }
        delete[] bptr;
    }
};

int main()
{
    DynamicMatrix * jaggerArr = new DynamicMatrix(4,5);
```

```
    jaggerArr->resizeMatrix(3,10);
    return 0;
}
```

```
printing without increment
0 1 2 3 4
1 2 3 4 5
2 3 4 5 6
3 4 5 6 7

printing without increment
0 1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9 10
2 3 4 5 6 7 8 9 10 11


--------------------------------
Process exited after 0.09761 seconds with return value 0
Press any key to continue . . .
```

Task 2:

matrix_multiply.h

```
#include<iostream>
namespace N{
void multiplyArray(int arr1[][3], int arr2[][3]);

}
```

matrix_multiply.cpp

matrix_multiply.cpp > multiplyArray(int [][3], int [][3])

```cpp
#include "matrix_multiply.h"

using namespace N;

void N::multiplyArray(int arr1[][3], int arr2[][3])
{
    // considering both arrays of same size
    int res[3][3];

    for (int i = 0; i < 3; ++i)
    {
        for (int j = 0; j < 3; ++j)
        {
            res[i][j] = 0;
        }
    }
    for (int i = 0; i < 3; ++i)
    {
        for (int j = 0; j < 3; ++j)
        {
            for (int k = 0; k < 3; ++k)
            {
                res[i][j] += arr1[i][k] * arr2[k][j];
            }
        }
    }
    std::cout << "displaying the resultant: " << std::endl;

    for (int i = 0; i < 3; ++i)
    {
        for (int j = 0; j < 3; ++j)
        {
            std::cout << res[i][j] << " ";
        }
        std::cout << std::endl;
    }
}
```

```cpp
#include <bits/stdc++.h>
#include "matrix_multiply.cpp"
using namespace std;

int main()
{
    int arr1[][3] = {
        {2, 3, 4},
        {5, 6, 7},
        {9, 5, 3}};
    int arr2[][3] = {
        {2, 2, 4},
        {6, 6, 7},
        {2, 6, 3}};
    cout << "Printing arr1: "
         << endl;
    for (int i = 0; i < 3; ++i)
    {
        for (int j = 0; j < 3; ++j)
        {
            cout << arr1[i][j] << " ";
        }
        cout << endl;
    }
    cout << endl
         << "Printing arr2: " << endl;
    for (int i = 0; i < 3; ++i)
    {
        for (int j = 0; j < 3; ++j)
        {
            cout << arr2[i][j] << " ";
        }
        cout << endl;
    };
    multiplyArray(arr1, arr2);
    return 0;
}
```

```
Printing arr1:
2 3 4
5 6 7
9 5 3

Printing arr2:
2 2 4
6 6 7
2 6 3
displaying the resultant:
30 46 41
60 88 83
54 66 80
```

Task 3:

```cpp
#include <bits/stdc++.h>

using namespace std;

class friendsArr
{
public:
    bool **basePtr;
    // implementing 3 rule
    friendsArr()
    {
        basePtr = new bool *[5];
        for (int i = 0; i < 5; ++i)
        {
            basePtr[i] = new bool[5];
        }
        for (int i = 0; i < 5; ++i)
        {
            for (int j = 0; j < 5; ++j)
            {
                basePtr[i][j] = false;
            }
        }
        basePtr[0][1] = true;
        basePtr[0][3] = true;
        basePtr[0][4] = true;
        basePtr[1][0] = true;
        basePtr[1][2] = true;
        basePtr[1][4] = true;
        basePtr[2][1] = true;
        basePtr[3][0] = true;
        basePtr[3][4] = true;
        basePtr[4][0] = true;
        basePtr[4][1] = true;
        basePtr[4][3] = true;
        for (int i = 0; i < 5; ++i)
        {
```

```
            for (int j = 0; j < 5; ++j)
            {
                cout << *(*(basePtr + i) + j) << " ";
            }
            cout << endl;
        }
    }

    void checkFriends(int a, int b)
    {
        bool flag = false;
        for (int k = 0; k < 5; ++k)
        {
            if (basePtr[a][k] == true && basePtr[b][k] == true)
            {

                cout << a << " and " << b << " have a common friend
at " << k << endl;
                flag = true;
            }
        }
        if (!flag)
            cout << "They do not have any common friends" << endl;
    }

    ~friendsArr()
    {
        for (int i = 0; i < 5; ++i)
        {
            delete[] basePtr[i];
        }
        delete[] basePtr;
    }
};
int main()
{
    friendsArr *ptr = new friendsArr();
    ptr->checkFriends(0, 4);
```

```
    // please enter your desired number to check all the cases
    ptr->checkFriends(1, 2);
    ptr->checkFriends(1, 3);
    return 0;
}
```

```
0 1 0 1 1
1 0 1 0 1
0 1 0 0 0
1 0 0 0 1
1 1 0 1 0
0 and 4 have a common friend at 1
0 and 4 have a common friend at 3
They do not have any common friends
1 and 3 have a common friend at 0
1 and 3 have a common friend at 4
```

Task 4:

```
#include <bits/stdc++.h>

using namespace std;

class GPAdata
{
public:
    double **baseptr;
    int courses[4] = {3, 4, 2, 1};
    string coursesName[4] = {"Software Engineering (SE) ",
"Artificial Intelligence (AI)", "Computer Science (CS), Data
Science (DS)"};
    GPAdata()
    {
        baseptr = new double *[4];
        for (int i = 0; i < 4; ++i)
        {
            baseptr[i] = new double[courses[i]];
        }
    }
    void setData()
    {
```

```
        for (int i = 0; i < 4; ++i)
        {
            for (int j = 0; j < courses[i]; ++j)
            {
                cout << "Enter Data for " << coursesName[i] << "
dep; course no " << j << endl;
                cin >> baseptr[i][j];
            }
        }
    }
    void showData()
    {
        for (int i = 0; i < 4; ++i)
        {
            for (int j = 0; j < courses[i]; ++j)
            {
                cout << baseptr[i][j] << " ";
            }
            cout << endl;
        }
    }
    ~GPAdata()
    {
        for (int i = 0; i < 4; ++i)
        {
            delete[] baseptr[i];
        }
        delete[] baseptr;
    }
};
int main()
{
    GPAdata *GPA1 = new GPAdata();
    GPA1->setData();
    GPA1->showData();
    return 0;
}
```

```
Enter Data for Software Engineering (SE)  dep; course no 0
3.5
Enter Data for Software Engineering (SE)  dep; course no 1
2.8
Enter Data for Software Engineering (SE)  dep; course no 2
2.0
Enter Data for Artificial Intelligence (AI) dep; course no 0
3.0
Enter Data for Artificial Intelligence (AI) dep; course no 1
4.0
Enter Data for Artificial Intelligence (AI) dep; course no 2
1.3
Enter Data for Artificial Intelligence (AI) dep; course no 3
2.5
Enter Data for Computer Science (CS), Data Science (DS) dep; course no 0
64.5
Enter Data for Computer Science (CS), Data Science (DS) dep; course no 1
3.4
Enter Data for  dep; course no 0
3.2
3.5 2.8 2
3 4 1.3 2.5
64.5 3.4
3.2

--------------------------------
Process exited after 45.34 seconds with return value 0
Press any key to continue . . .
```

Task 5:

```cpp
#include <bits/stdc++.h>

using namespace std;
// I will be using jagged array sttucture to solve this question
class seatManagmentSystem
{
public:
    string **baseptr;
    int rows;
    int *columns;

    seatManagmentSystem(int rows)
    {
        this->rows = rows;
        baseptr = new string *[rows];
        columns = new int[rows];
```

```
        for (int i = 0; i < rows; ++i)
        {
            cout << "Enter number of seats in row " << i + 1 <<
endl;
            cin >> columns[i];
        }
        for (int i = 0; i < rows; ++i)
        {
            baseptr[i] = new string[columns[i]];
        }
    }

    void setSeatNames()
    {
        fflush(stdin);
        for (int i = 0; i < rows; ++i)
        {
            for (int j = 0; j < columns[i]; ++j)
            {
                cout << "Enter name for seat " << j + 1 << " in row
" << i + 1 << endl;
                getline(cin, baseptr[i][j]);
            }
        }
    }
    void displaySeatChart()
    {
        for (int i = 0; i < rows; ++i)
        {
            for (int j = 0; j < columns[i]; ++j)
            {
                cout << "name for seat " << j + 1 << " in row " <<
i + 1 << " ";
                cout << baseptr[i][j] << endl;
            }
            cout << endl;
        }
    }
```

```
    ~seatManagmentSystem()
    {
        for (int i = 0; i < rows; ++i)
        {
            delete[] baseptr;
        }
        delete[] baseptr;
    }
};
int main()
{
    seatManagmentSystem *ptr = new seatManagmentSystem(4);
    ptr->setSeatNames();
    ptr->displaySeatChart();
    return 0;
}
```

```
Enter number of seats in row 1
3
Enter number of seats in row 2
4
Enter number of seats in row 3
1
Enter number of seats in row 4
2
Enter name for seat 1 in row 1
ahmed ali
Enter name for seat 2 in row 1
anwar
Enter name for seat 3 in row 1
mufeez hanif
Enter name for seat 1 in row 2
someone
Enter name for seat 2 in row 2
none
Enter name for seat 3 in row 2
any one
Enter name for seat 4 in row 2
shoab
Enter name for seat 1 in row 3
jaseem ali
Enter name for seat 1 in row 4
misbah ahmed
Enter name for seat 2 in row 4
kashan hussain shah
name for seat 1 in row 1 ahmed ali
name for seat 2 in row 1 anwar
name for seat 3 in row 1 mufeez hanif

name for seat 1 in row 2 someone
name for seat 2 in row 2 none
name for seat 3 in row 2 any one
name for seat 4 in row 2 shoab

name for seat 1 in row 3 jaseem ali

name for seat 1 in row 4 misbah ahmed
name for seat 2 in row 4 kashan hussain shah
```