#include <iostream>

#include<windows.h>

#include <iomanip>

using namespace std;

// 2D Task

int\*\* ReserveArray(const int& row, const int& col)

{

int\*\* arr = new int\* [row] {};

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

{

arr[x] = new int[col] {};

}

return arr;

}

void InitArray(int\*\*& arr, const int& row, const int& col)

{

srand(time(0));

int min = 1;

int max = 10;

int random = 0;

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

{

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

{

random = min + rand() % (max - min);

arr[r][c] = random;

}

}

}

void colorStart()

{

HANDLE consolehwnd = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute(consolehwnd, FOREGROUND\_GREEN);

}

void colorEnd()

{

HANDLE consolehwnd = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute(consolehwnd, FOREGROUND\_INTENSITY);

}

void PrintArray(int\*\*& arr, const int& row, const int& col)

{

colorStart();

cout << "\n My 2D Array : " << endl;

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

{

if (r == 0)

{

cout << " ";

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

{

cout << "\_\_\_\_\_\_\_";

}

cout << endl;

}

else

{

cout << " ";

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

{

cout << "|\_\_\_\_\_\_";

}

cout << "|" << endl;

}

cout << " ";

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

{

cout << "|" << setw(6) << arr[r][c];

}

cout << "|" << endl;

}

cout << " ";

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

{

cout << "|\_\_\_\_\_\_";

}

cout << "|" << endl;

colorEnd();

}

// Task 1

// Matrixi X oxu üzrə qatlayın və üst-üstə düşən elementləri vurun

int\*\* FoldEvenMatrixX(int\*\*& arr, int& row, const int& col)

{

auto newArray = ReserveArray(row / 2 , col);

for (int r = 0; r < row / 2; r++)

{

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

{

newArray[r][c] = arr[r][c] + arr[row - r - 1][c];

}

}

row = row / 2;

return newArray;

}

int\*\* FoldOddMatrixX(int\*\*& arr, int& row, const int& col, int& rowCopy)

{

auto newArray = ReserveArray(row / 2 + 1, col);

for (int r = 0; r < row / 2; r++)

{

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

{

newArray[r][c] = arr[r][c] + arr[row - r - 1][c];

}

}

if (row == rowCopy)

{

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

{

newArray[row / 2][x] = arr[rowCopy / 2][x];

}

row = row / 2 + 1;

rowCopy = rowCopy / 2 + 1;

return newArray;

}

else if (row == 3)

{

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

{

newArray[row / 2][x] = arr[row / 2][x];

}

row = row / 2 + 1;

return newArray;

}

else

{

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

{

newArray[row / 2 ][x] = arr[row / 2][x];

}

row = row / 2 + 1;

return newArray;

}

}

// Task 2

// Matrixi Y oxu üzrə qatlayın və üst-üstə düşən elementləri toplayın

int\*\* FoldEvenMatrixY(int\*\*& arr, const int& row, int& col) {

auto newArray = ReserveArray(row, col / 2);

for (int c = 0; c < col / 2; c++)

{

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

{

newArray[r][c] = arr[r][c] + arr[r][col - c - 1];

}

}

col = col / 2;

return newArray;

}

int\*\* FoldOddMatrixY(int\*\*& arr, const int& row, int& col, int& colCopy)

{

auto newArray = ReserveArray(row, col / 2 + 1);

for (int c = 0; c < col / 2; c++)

{

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

{

newArray[r][c] = arr[r][c] + arr[r][col - c - 1];

}

}

if (col == colCopy)

{

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

{

newArray[x][col / 2] = arr[x][colCopy / 2];

}

col = col / 2 + 1;

colCopy = colCopy / 2 + 1;

return newArray;

}

else if (col == 3)

{

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

{

newArray[x][col / 2] = arr[x][col / 2 + 1];

}

col = col / 2 + 1;

return newArray;

}

else

{

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

{

newArray[x][col / 2] = arr[x][col / 2];

}

col = col / 2 + 1;

return newArray;

}

}

// Task 3

// Sütunun minimal ölçüyə gələnə qədər sıxışdırın

int\*\* MinimizeEvenArrayY(int\*\*& arr, const int& row, int& col) {

auto newArray = ReserveArray(row, col / 2);

for (int c = 0; c < col / 2; c++)

{

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

{

newArray[r][c] = arr[r][c] + arr[r][col - c - 1];

}

}

col = col / 2;

return newArray;

}

int\*\* MinimizeOddArrayY(int\*\*& arr, const int& row, int& col, int& colCopy)

{

auto newArray = ReserveArray(row, col / 2 + 1);

for (int c = 0; c < col / 2; c++)

{

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

{

newArray[r][c] = arr[r][c] + arr[r][col - c - 1];

}

}

if (col == colCopy)

{

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

{

newArray[x][col / 2] = arr[x][colCopy / 2];

}

col = col / 2 + 1;

colCopy = colCopy / 2 + 1;

return newArray;

}

else if (col == 3)

{

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

{

newArray[x][col / 2] = arr[x][col / 2 + 1];

}

col = col / 2 + 1;

return newArray;

}

else

{

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

{

newArray[x][col / 2] = arr[x][col / 2];

}

col = col / 2 + 1;

return newArray;

}

}

// Task 4

// Sətiri minimal ölçüyə gələnə qədər sıxışdırın

int\*\* MinimizeEvenArrayX(int\*\*& arr, int& row, const int& col)

{

auto newArray = ReserveArray(row / 2, col);

for (int r = 0; r < row / 2; r++)

{

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

{

newArray[r][c] = arr[r][c] + arr[row - r - 1][c];

}

}

row = row / 2;

return newArray;

}

int\*\* MinimizeOddArrayX(int\*\*& arr, int& row, const int& col, int& rowCopy)

{

auto newArray = ReserveArray(row / 2 + 1, col);

for (int r = 0; r < row / 2; r++)

{

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

{

newArray[r][c] = arr[r][c] + arr[row - r - 1][c];

}

}

if (row == rowCopy)

{

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

{

newArray[row / 2][x] = arr[rowCopy / 2][x];

}

row = row / 2 + 1;

rowCopy = rowCopy / 2 + 1;

return newArray;

}

else if (row == 3)

{

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

{

newArray[row / 2][x] = arr[row / 2 + 1][x];

}

row = row / 2 + 1;

return newArray;

}

else

{

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

{

newArray[row / 2][x] = arr[row / 2][x];

}

row = row / 2 + 1;

return newArray;

}

}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

void line()

{

colorStart();

cout << "\n========================================================================================================================" << endl;

colorEnd();

}

void cn()

{

colorStart();

int space = 0;

cout << "\n TO GO ANOTHER TASK, PRESS 1 . . . ";

cin >> space;

if (space == 1)

system("cls");

colorEnd();

}

void main()

{

void (\*line\_ptr)() = line;

void (\*cn\_ptr)() = cn;

colorStart();

// Task 1

line\_ptr();

colorStart();

cout << "\n TASK 1" << endl;

int row = 0, col = 0;

cout << "\n Enter the number of rows : ";

cin >> row;

cout << "\n Enter the number of columns : ";

cin >> col;

colorEnd();

auto arr = ReserveArray(row, col);

InitArray(arr, row, col);

PrintArray(arr, row, col);

int rowCopy = row;

if (row % 2 == 0)

{

arr = FoldEvenMatrixX(arr, row, col);

PrintArray(arr, row, col);

}

else

{

arr = FoldOddMatrixX(arr, row, col, rowCopy);

PrintArray(arr, row, col);

}

line\_ptr();

cn();

colorStart();

// Task 2

line\_ptr();

colorStart();

cout << "\n TASK 2" << endl;

int row2 = 0, col2 = 0;

cout << "\n Enter the number of rows : ";

cin >> row2;

cout << "\n Enter the number of columns : ";

cin >> col2;

colorEnd();

auto arr2 = ReserveArray(row2, col2);

InitArray(arr2, row2, col2);

PrintArray(arr2, row2, col2);

int colCopy = col2;

if (col2 % 2 == 0)

{

arr2 = FoldEvenMatrixY(arr2, row2, col2);

PrintArray(arr2, row2, col2);

}

else

{

arr2 = FoldOddMatrixY(arr2, row2, col2, colCopy);

PrintArray(arr2, row2, col2);

}

line\_ptr();

cn();

colorStart();

// Task 3

line\_ptr();

colorStart();

cout << "\n TASK 3" << endl;

int row3 = 0, col3 = 0;

cout << "\n Enter the number of rows : ";

cin >> row3;

cout << "\n Enter the number of columns : ";

cin >> col3;

colorEnd();

auto arr3 = ReserveArray(row3, col3);

InitArray(arr3, row3, col3);

PrintArray(arr3, row3, col3);

if (col3 % 2 == 0)

{

while (col3 != 1)

{

int colCopy2 = col3;

if (col3 % 2 == 0)

{

arr3 = MinimizeEvenArrayY(arr3, row3, col3);

PrintArray(arr3, row3, col3);

}

else

{

arr3 = MinimizeOddArrayY(arr3, row3, col3, colCopy2);

PrintArray(arr3, row3, col3);

}

}

}

else

{

while (col3 != 1)

{

int colCopy2 = col3;

if (col3 % 2 == 1)

{

arr3 = MinimizeOddArrayY(arr3, row3, col3, colCopy2);

PrintArray(arr3, row3, col3);

}

else

{

arr3 = MinimizeEvenArrayY(arr3, row3, col3);

PrintArray(arr3, row3, col3);

}

}

}

line\_ptr();

cn();

colorStart();

// Task 4

line\_ptr();

colorStart();

cout << "\n TASK 4" << endl;

int row4 = 0, col4 = 0;

cout << "\n Enter the number of rows : ";

cin >> row4;

cout << "\n Enter the number of columns : ";

cin >> col4;

colorEnd();

auto arr4 = ReserveArray(row4, col4);

InitArray(arr4, row4, col4);

PrintArray(arr4, row4, col4);

if (row4 % 2 == 0)

{

while (row4 != 1)

{

int rowCopy2 = row4;

if (row4 % 2 == 1)

{

arr4 = MinimizeOddArrayX(arr4, row4, col4,rowCopy2);

PrintArray(arr4, row4, col4);

}

else

{

arr4 = MinimizeEvenArrayX(arr4, row4, col4);

PrintArray(arr4, row4, col4);

}

}

}

else

{

while (row4 != 1)

{

int rowCopy2 = row4;

if (row4 % 2 == 0)

{

arr4 = MinimizeEvenArrayX(arr4, row4, col4);

PrintArray(arr4, row4, col4);

}

else

{

arr4 = MinimizeOddArrayX(arr4, row4, col4, rowCopy2);

PrintArray(arr4, row4, col4);

}

}

}

line\_ptr();

cn();

line\_ptr();

colorStart();

cout << "\n Tasks Ended." << endl;

line\_ptr();

colorEnd();

}