Trạng thái	Đã xong
Bắt đầu vào lúc	Thứ Năm, 17 tháng 4 2025, 12:08 PM
Kết thúc lúc	Thứ Năm, 17 tháng 4 2025, 12:48 PM
Thời gian thực hiện	40 phút 49 giây

Câu hỏi **1**

Đúng

Đạt điểm 1,00

Mô tả tiếng Việt:

Cho mảng 2 chiều chứa các số nguyên, kích thước M x N.

Hiện thực hàm:

```
int findMaxColumn(int arr[][1000], int row, int col);
```

Trong đó; `arr`, `row` và `col` lần lượt là mảng 2 chiều, số hàng và số cột của mảng. Tìm chỉ số của cột có tổng tất cả các phần tử lớn nhất.

Lưu ý: Cột đầu tiên được đánh chỉ số 0. Nếu có nhiều hơn một cột có tổng lớn nhất, ta chọn cột có chỉ số lớn nhất.

Ghi chú: (Các) thư viện `iostream` và `climits` đã được khai báo, và `namespace std` đã được sử dụng.

English version:

Given a two-dimensional array whose each element is integer, its size is M x N.

Implement the following function:

```
int findMaxColumn(int arr[][1000], int row, int col);
```

Where `arr`, `row` and `col` are the given two-dimensional array, its number of rows and its number of columns. Find the index of the column which has the greatest sum of all elements on it.

Note: The first column of the given array is numbered by 0. If there are more than one column whose sum is the greatest, choose the column with the greatest index.

Note: Libraries `iostream` and `climits` have been imported, and `namespace std` has been used.

For example:

Test	Result
`int arr[][1000] = {{-44,64,-6},{87,92,-19},{-92,53,-38},{-39,-92,21}};` `cout << findMaxColumn(arr, 4, 3);`	1
`int arr[][1000] = {{-92,78,-2,-58,-37},{44,-4,30,-69,22}};` `cout << findMaxColumn(arr, 2,5);`	1

Answer: (penalty regime: 0 %)

Reset answer

```
int findMaxColumn(int arr[][1000], int row, int c
    int max = INT_MIN;
    int res = 0;

    for(int i = 0; i < col; ++i){
        int sum = 0;
        for(int j = 0; j < row; ++j){
            sum += arr[j][i];
            }
        if(sum >= max){
                max = sum;
                res = i;
            }
```

```
    }
    return res;
}
```

	Test	Expected	Got	
✓	`int arr[][1000] = {{-44,64,-6},{87,92,-19}, {-92,53,-38},{-39,-92,21}}; cout << findMaxColumn(arr, 4, 3);`	1	1	✓
✓	`int arr[][1000] = {{-92,78,-2,-58,-37}, {44,-4,30,-69,22}}; cout << findMaxColumn(arr, 2,5);`	1	1	✓

Passed all tests! ✓

Câu hỏi **2**

Đúng

Đạt điểm 1,00

Mô tả tiếng Việt:

Cho mảng 2 chiều chứa các số nguyên, kích thước N x N.

Hiện thực hàm:

```
int diagonalProduct(int arr[][1000], int row, int col);
```

Trong đó; `arr`, `row` và `col` lần lượt là mảng 2 chiều, số hàng và số cột của mảng. Tìm tích của tất cả các phần tử trong đường chéo chính của mảng.

Ghi chú: (Các) thư viện `iostream`, và `string` đã được khai báo, và `namespace std` đã được sử dụng.

English version:

Given a two-dimensional array whose each element is integer, its size is N x N.

Implement the following function:

```
int diagonalProduct(int arr[][1000], int row, int col);
```

Where `arr`, `row` and `col` are the given two-dimensional array, its number of rows and its number of columns. Find the product of all elements on the main diagonal of this array.

Note: Libraries `iostream`, and `string` have been imported, and `namespace std` has been used.

For example:

Test	Result
`int arr[][1000] = {{-45,18,-37},{-2,-31,24}, {-48,-33,-48}}; cout << diagonalProduct(arr,3,3);`	-66960
`int arr[][1000] = {{-11,44,18,33}, {-34,-9,-42,-42},{47,-26,4,-8},{-35,11,-34,-19}}; cout << diagonalProduct(arr,4,4);`	-7524

Answer: (penalty regime: 0 %)

Reset answer

```
int diagonalProduct(int arr[][1000], int row, int
    int res = 1;
    for(int i = 0; i < row; ++i){
        res *= arr[i][i];
    }
    return res;
}
```

	Test	Expected	Got	
✓	int arr[][1000] = {{-45,18,-37},{-2,-31,24},{-48,-33,-48}}; cout << diagonalProduct(arr,3,3);	-66960	-66960	✓
✓	int arr[][1000] = {{-11,44,18,33},{-34,-9,-42,-42},{47,-26,4,-8},{-35,11,-34,-19}}; cout << diagonalProduct(arr,4,4);	-7524	-7524	✓

Passed all tests! ✓

Câu hỏi **3**

Đúng

Đạt điểm 1,00

Mô tả tiếng Việt:

Cho mảng 2 chiều chứa các số nguyên, kích thước N x N.

Hiện thực hàm:

```
bool isSymmetric(int arr[][1000], int row, int col);
```

Trong đó; `arr`, `row` và `col` lần lượt là mảng 2 chiều, số hàng và số cột của mảng. Một ma trận được gọi là đối xứng nếu với mọi i, j; giá trị của phần tử ở hàng i, cột j luôn bằng giá trị của phần tử ở hàng j, cột i. Kiểm tra xem mảng này có phải là một ma trận đối xứng hay không; trả về **true** nếu mảng này là ma trận đối xứng; ngược lại, trả về **false**.

Ghi chú: (Các) thư viện `iostream` và `string` đã được khai báo, và `namespace std` đã được sử dụng.

English version:

Given a two-dimensional array whose each element is integer, its size is N x N.

Implement the following function:

```
bool isSymmetric(int arr[][1000], int row, int col);
```

Where `arr`, `row` and `col` are the given two-dimensional array, its number of rows and its number of columns. A matrix is called as **symmetric matrix** if for all i, j; the value of the element on row i, column j is equal to the value of the element on row j, column i. Check whether the given array is **symmetric matrix** or not; return **true** if it is, otherwise return **false**.

Note: Libraries `iostream` and `string` have been imported, and `namespace std` has been used.

For example:

Test	Result
`int arr[][1000] = {{1,4,6}, {4,5,3}, {6,3,9}};` `cout << isSymmetric(arr,3,3);`	1
`int arr[][1000] = {{1,9,6}, {4,5,3}, {6,3,9}};` `cout << isSymmetric(arr,3,3);`	0

Answer: (penalty regime: 0 %)

Reset answer

```
bool isSymmetric(int arr[][1000], int row, int co
    for(int i = 0; i < row; ++i){
        for(int j = 0; j < col; ++j){
            if(arr[i][j] != arr[j][i])
                return false;
        }
    }
    return true;
}
```

	Test	Expected	Got	
✓	`int arr[][1000] = {{1,4,6}, {4,5,3}, {6,3,9}};` `cout << isSymmetric(arr,3,3);`	1	1	✓
✓	`int arr[][1000] = {{1,9,6}, {4,5,3}, {6,3,9}};` `cout << isSymmetric(arr,3,3);`	0	0	✓

Passed all tests! ✓

Câu hỏi **4**

Đúng

Đạt điểm 1,00

Mô tả tiếng Việt:

Cho mảng 2 chiều chứa các số nguyên, kích thước M x N.

Hiện thực hàm:

```
int diagonalDiff(int arr[][1000], int row, int col, int x, int y);
```

Trong đó; `arr`, `row` và `col` lần lượt là mảng 2 chiều, số hàng, số cột của mảng; `x` và `y` biểu thị ô có số hàng là x và số cột là y trong mảng đã cho (`0≤x<row` và `0≤y<col`). Tổng của một đường chéo là tổng tất cả các phần tử nằm trên đường chéo đó. Tìm giá trị tuyệt đối của hiệu giữa hai đường chéo đi qua ô có số hàng x và số cột y.

Ghi chú: (Các) thư viện `iostream`, `vector` và `string` đã được khai báo, và `namespace std` đã được sử dụng.

English version:

Given a two-dimensional array whose each element is an integer, its size is M x N.

Implement the following function:

```
int diagonalDiff(int arr[][1000], int row, int col, int x, int y);
```

Where `arr`, `row` and `col` are the given two-dimensional array, its number of rows and its number of columns. ; `x` and `y` represent the cell whose index of the row is x and index of the column is y (`0≤x<row` và `0≤y<col`). The sum of a diagonal is the sum of all elements on it. Find the absolute value of the difference between the sums of two diagonal containing the cell which is represented by x and y of the given array.

Note: Libraries `iostream`, `vector`, and `string` have been imported, and `namespace std` has been used.

For example:

Test	Result
`int arr[][1000] = {{55,85,47,58},{31,4,60,67},{94,69,71,73},{51,62,64,90}};` `cout << diagonalDiff(arr,4,4,1,2);`	20
`int arr[][1000] = {{88,72,65,37},{82,84,34,12},{74,46,88,44}};` `cout << diagonalDiff(arr,3,4,1,0);`	26

Answer: (penalty regime: 0 %)

Reset answer

```
int diagonalDiff(int arr[][1000], int row, int co
    int sum1 = 0;
    int sum2 = 0;
    for(int i = 0; i < row; ++i){
        for(int j = 0; j < col; ++j)
        {
            if(x + y == i + j)
                sum1 += arr[i][j];
            if(x - y == i - j)
                sum2 += arr[i][j];
        }
    }
    return (sum1 - sum2 > 0) ? sum1 - sum2 : -(su
}
// .    .   .   .   .
// .    .   .   .   .
// .    .   .   .   .
// .    .   .   .   .
```

	Test	Expected	Got	
✓		20	20	✓

	`int arr[][1000] = {{55,85,47,58},{31,4,60,67},{94,69,71,73},{51,62,64,90}}; cout << diagonalDiff(arr,4,4,1,2);`			
✓	`int arr[][1000] = {{88,72,65,37},{82,84,34,12},{74,46,88,44}}; cout << diagonalDiff(arr,3,4,1,0);`	26	26	✓

Passed all tests! ✓