Trạng thái	Đã xong
Bắt đầu vào lúc	Thứ Tư, 16 tháng 4 2025, 4:45 PM
Kết thúc lúc	Thứ Tư, 16 tháng 4 2025, 5:04 PM
Thời gian thực hiện	18 phút 21 giây

Câu hỏi 1

Đúng

Mô tả tiếng Việt:

Hãy hiện thực hàm **int* zeros(int n)** tạo một mảng có n phần tử 0.

Đầu vào: Kích thước mảng n.

Đầu ra: Con trỏ trỏ tới mảng vừa được cấp phát.

Lưu ý: Trong trường hợp cấp phát thất bại, hàm sẽ trả về nullptr.

English version:

Implement the function **int* zeros(int n)** which can create an array with n zero element.

Input: The array size n.

Output: The pointer that points to the allocated array.

Note: In the case of failed allocation, the function will return nullptr value.

For example:

Test	Input	Result
1	1	0

Answer: (penalty regime: 0 %)

Reset answer

```
int* zeros(int n) {
    // Kiểm tra nếu n không hợp lệ (n <= 0), trả về nullptr
    if (n <= 0) {
        return nullptr;
    }

    // Cấp phát động một mảng n phần tử kiểu int
    // Sử dụng std::nothrow để tránh ném ngoại lệ khi cấp phát thất bại
    int* arr = new (std::nothrow) int[n];

    // Kiểm tra xem việc cấp phát có thành công không
    // Nếu thất bại (arr == nullptr), trả về nullptr
    if (arr == nullptr) {
        return nullptr;
    }

    // Khởi tạo tất cả phần tử của mảng bằng 0
    for (int i = 0; i < n; i++) {
        arr[i] = 0;
    }

    // Trả về con trỏ trỏ đến mảng đã được cấp phát và khởi tạo
    return arr;

    // Lưu ý: Người sử dụng hàm này cần phải giải phóng bộ nhớ bằng delete[] arr
    // sau khi sử dụng xong để tránh rò rỉ bộ nhớ
}
```

	Test	Input	Expected	Got	
✓	1	1	0	0	✓

Passed all tests! ✓

Câu hỏi **2**

Đúng

Mô tả tiếng Việt:

Hãy hiện thực hàm **void shallowCopy(int*& newArr, int*& arr)** có chức năng tạo một bản sao của một mảng một chiều.

Đầu vào: Mảng một chiều arr cần được sao chép.

Đầu ra: Mảng đích một chiều newArr cần sao chép tới.

Lưu ý: sau thực thi mảng được sao chép và mảng cần sao chép đều sử dụng chung một vùng nhớ.

English version:

Implement the function **void shallowCopy(int*& newArr, int*& arr)** that can create a copy from a one-dimensional array.

Input: The one-dimensional array that needs to be copied.

Output: The destination array.

Note: After finishing execution, both the one-dimensional array that needs to be copied and the destination array use the same data memory.

For example:

Test	Result

`int* arr = new int[2];`
`arr[0] = 2; arr[1] = 3;`
`int* newArr = nullptr;`
`shallowCopy(newArr, arr);`
`cout << newArr[0] << ' ' << newArr[1];`
`delete[] arr;`
	2 3

Answer: (penalty regime: 0 %)

Reset answer

```
void shallowCopy(int*& newArr, int*& arr) {
    // Chỉ cần gán con trỏ newArr bằng con trỏ arr
    // Điều này làm cho cả hai con trỏ cùng trỏ đến cùng một vùng nhớ
    newArr = arr;

    // Lưu ý: Đây là shallow copy (sao chép nông)
    // Cả hai con trỏ newArr và arr đều trỏ đến cùng một mảng dữ liệu
    // Thay đổi dữ liệu thông qua một con trỏ sẽ ảnh hưởng đến con trỏ còn lại
    // Khi giải phóng bộ nhớ, chỉ nên sử dụng delete[] trên một trong hai con trỏ
}
```

	Test	Expected	Got	

✓	int* arr = new int[2];
	arr[0] = 2; arr[1] = 3;
	int* newArr = nullptr;
	shallowCopy(newArr, arr);
	cout << newArr[0] << ' ' << newArr[1];
	delete[] arr;
		2 3	2 3	✓

✓	int* arr = new int[8];
	arr[6] = 2; arr[6] = 5;
	int* newArr = nullptr;
	shallowCopy(newArr, arr);
	arr[6] = 40;
	cout << newArr[6];
	delete[] arr;
		40	40	✓

Passed all tests! ✓

Câu hỏi **3**

Đúng

Mô tả tiếng Việt:

Hãy hiện thực hàm **int** deepCopy(int** matrix, int r, int c)** trả về một bản sao của matrix gồm r hàng và n cột.

Đầu vào: Con trỏ matrix trỏ đến mảng hai chiều có kích thước r x c.

Đầu ra: Con trỏ trỏ đến mảng hai chiều được sao chép.

Lưu ý: sau thực thi, con trỏ trả về phải trỏ đến vùng nhớ được cấp phát mới và khi matrix truyền vào có kích thước 0, hàm trả về nullptr.

English version:

Implement the function **int** deepCopy(int** matrix, int r, int c)** that return a copy of a matrix consisting of r rows and c colmuns.

Input: Pointer arr points to the one-dimensional array that needs to be copied.

Output: Pointer newArr points to the destination array.

Note: After finishing execution, the one-dimensional array that needs to be copied and the destination array use the two distinct data memory.

For example:

Test	Result

`int** m = new int*[2];`
`m[0] = new int[2]; m[0][0] = 1; m[0][1] = 2;`
`m[1] = new int[2]; m[1][0] = 1; m[1][1] = 3;`
`int** n = deepCopy(m, 2, 2);`
`cout << n[0][0] << ' ' << n[0][1] << '\n' << n[1][0] << ' ' << n[1][1];`
	1 2
	1 3

Answer: (penalty regime: 0 %)

Reset answer

```
int** deepCopy(int** matrix, int r, int c) {
    // Kiểm tra nếu kích thước ma trận không hợp lệ
    if (r <= 0 || c <= 0 || matrix == nullptr) {
        return nullptr;
    }

    // Cấp phát mảng các con trỏ cho các hàng của ma trận mới
    int** newMatrix = new (std::nothrow) int*[r];
    if (newMatrix == nullptr) {
        return nullptr;
    }

    // Cấp phát và sao chép từng hàng của ma trận
    for (int i = 0; i < r; i++) {
        // Cấp phát bộ nhớ cho hàng thứ i
        newMatrix[i] = new (std::nothrow) int[c];

        // Nếu cấp phát thất bại, giải phóng bộ nhớ đã cấp phát trước đó và trả về nullptr
        if (newMatrix[i] == nullptr) {
            // Giải phóng bộ nhớ đã được cấp phát cho các hàng trước đó
            for (int j = 0; j < i; j++) {
                delete[] newMatrix[j];
            }
            delete[] newMatrix;
            return nullptr;
        }

        // Sao chép dữ liệu từ ma trận gốc sang ma trận mới
        for (int j = 0; j < c; j++) {
            newMatrix[i][j] = matrix[i][j];
        }
    }

    return newMatrix;
}
```

	Test	Expected	Got	

✓	int** m = new int*[2];
	m[0] = new int[2]; m[0][0] = 1; m[0][1] = 2;
	m[1] = new int[2]; m[1][0] = 1; m[1][1] = 3;
	int** n = deepCopy(m, 2, 2);
	cout << n[0][0] << ' ' << n[0][1] << '\n' << n[1][0] << ' ' << n[1][1];
		1 2
		1 3
			1 2
			1 3
				✓

✓	int** m = new int*[2];
	m[0] = new int[2]; m[0][0] = 6; m[0][1] = 2;
	m[1] = new int[2]; m[1][0] = 8; m[1][1] = 3;
	int** n = deepCopy(m, 2, 2);
	cout << n[0][0] << ' ' << n[0][1] << '\n' << n[1][0] << ' ' << n[1][1];
		6 2
		8 3
			6 2
			8 3
				✓

Passed all tests! ✓

Câu hỏi **4**

Đúng

Mô tả tiếng Việt:

Hãy hiện thực hàm **void deleteMatrix(int**& matrix, int r)** thực hiện giải phóng ô nhớ cho một mảng động 2 chiều có r hàng. **matrix** được gán bằng giá trị NULL sau khi thực hiện hàm.

Đầu vào: Mảng động hai chiều matrix có số hàng r cần giải phóng ô nhớ.

English version:

Implement the function **void deleteMatrix(int**& matrix, int r)** that can free memory for a dynamic two-dimensional array consisting of r rows. **matrix** should be set to NULL after function's execution.

Input: The dynamic two-dimensional array, matrix, consists of r rows.

For example:

Test	Input	Result
1	2 2 1 1 1 1	SUCCESSFUL

Answer: (penalty regime: 0 %)

Reset answer

```
void deleteMatrix(int**& matrix, int r) {
    // Kiểm tra nếu matrix là nullptr hoặc r <= 0, không cần giải phóng
    if (matrix == nullptr || r <= 0) {
        // Đảm bảo matrix được đặt thành nullptr trước khi kết thúc
        matrix = nullptr;
        return;
    }

    // Giải phóng từng hàng của ma trận
    for (int i = 0; i < r; i++) {
        // Trước khi giải phóng, kiểm tra xem con trỏ hàng có tồn tại không
        if (matrix[i] != nullptr) {
            delete[] matrix[i];
        }
    }

    // Giải phóng mảng các con trỏ hàng
    delete[] matrix;

    // Đặt matrix thành nullptr để tránh dangling pointer
    matrix = nullptr;
}
```

	Test	Input	Expected	Got	
✓	1	2 2 1 1 1 1	SUCCESSFUL	SUCCESSFUL	✓
✓	10	0 1	SUCCESSFUL	SUCCESSFUL	✓

Passed all tests! ✓

Câu hỏi **5**

Đúng

Mô tả tiếng Việt:

Cho một mảng động hai chiều matrix có kích thước r x c. Hiện thực hàm **void insertRow(int**& matrix, int r, int c, int* rowArr, int row)** tiến hành chèn mảng rowArr (có kích thước c) vào hàng thứ row của mảng matrix.

Đầu vào: Mảng 2 chiều matrix có kích thước r x c, hàng cần chèn rowArr và vị trí chèn row.

Đầu ra: Mảng 2 chiều matrix sau khi được chèn.

English version:

Given a dynamic two-dimensional array of size r x c. Implement the function **void insertRow(int**& matrix, int r, int c, int* rowArr, int row)** that can insert the rowArr array (with the size c) into the row position, row, of the matrix.

Input: The two-dimensional matrix of size r x c, the insert row rowArr and the insert position row.

Output: The two-dimensional matrix after insert.

For example:

Test	Input	Result
1	2 3 1 2 3 4 5 6 2 7 8 9	1 2 3 4 5 6 7 8 9

Answer: (penalty regime: 0 %)

Reset answer

```
void insertRow(int**& matrix, int r, int c, int* rowArr, int row) {
    // Kiểm tra các điều kiện đầu vào
    if (matrix == nullptr || rowArr == nullptr || row < 0 || row > r) {
        return; // Không thực hiện thao tác nếu điều kiện không hợp lệ
    }

    // Cấp phát ma trận mới với (r+1) hàng và c cột
    int** newMatrix = new (std::nothrow) int*[r + 1];
    if (newMatrix == nullptr) {
        return; // Nếu không thể cấp phát bộ nhớ, thoát hàm
    }

    // Sao chép các hàng từ ma trận cũ tới ma trận mới
    // và chèn rowArr vào vị trí row
    for (int i = 0; i <= r; i++) {
        if (i == row) {
            // Chèn rowArr tại vị trí row
            newMatrix[i] = new (std::nothrow) int[c];
            if (newMatrix[i] == nullptr) {
                // Nếu cấp phát thất bại, giải phóng bộ nhớ đã cấp phát và thoát
                for (int j = 0; j < i; j++) {
                    delete[] newMatrix[j];
                }
                delete[] newMatrix;
                return;
            }

            // Sao chép dữ liệu từ rowArr vào hàng mới
            for (int j = 0; j < c; j++) {
                newMatrix[i][j] = rowArr[j];
            }
        } else if (i < row) {
            // Sao chép các hàng trước vị trí chèn
            newMatrix[i] = matrix[i];
        } else {
            // Sao chép các hàng sau vị trí chèn (dịch xuống 1 hàng)
            newMatrix[i] = matrix[i - 1];
```

```
        }
    }

    // Giải phóng mảng con trỏ cũ, nhưng giữ lại các hàng vì đã được chuyển sang ma trận mới
    delete[] matrix;

    // Cập nhật matrix để trỏ đến ma trận mới
    matrix = newMatrix;
}
```

	Test	Input	Expected	Got	
✓	1	2 3 1 2 3 4 5 6 2 7 8 9	1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9	✓

Passed all tests! ✓