Câu 1:

#### **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 |

int\* zeros(int n) {

int\* arr = new int[n]; // dynamically allocate an array of size n

if (arr == nullptr) {

// allocation failed, return nullptr

return nullptr;

}

for (int i = 0; i < n; i++) {

arr[i] = 0; // gán giá trị 0 cho từng phần tử của mảng

}

return arr;

}

Câu 2

**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 %)

void shallowCopy(int\*& newArr, int\*& arr) {

newArr = arr;

}

Câu 3: **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 %)

int\*\* deepCopy(int\*\* matrix, int r, int c) {

// Kiểm tra kích thước của ma trận đầu vào

if (r == 0 || c == 0) {

return nullptr;

}

// Cấp phát bộ nhớ cho ma trận mới

int\*\* copy = new int\*[r];

for (int i = 0; i < r; i++) {

copy[i] = new int[c];

}

// Sao chép giá trị từ ma trận đầu vào sang ma trận mới

for (int i = 0; i < r; i++) {

for (int j = 0; j < c; j++) {

copy[i][j] = matrix[i][j];

}

}

return copy;

}

Câu 4: **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 %)

void deleteMatrix(int\*\*& matrix, int r) {

for(int i = 0 ; i<r;i++){

delete[] matrix[i];

matrix[i] = nullptr;

}

delete[] matrix;

matrix = nullptr;

}

Câu 5:

**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 |

void insertRow(int\*\*& matrix, int r, int c, int\* rowArr, int row) {

// TODO

// Cấp phát bộ nhớ cho một mảng mới có kích thước r+1 x c

int\*\* tempMatrix = new int\*[r+1];

for (int i = 0; i < r+1; i++) {

tempMatrix[i] = new int[c];

}

// Sao chép các hàng từ mảng matrix vào mảng mới, trừ hàng tại vị trí row

for (int i = 0; i < r+1; i++) {

if (i < row) {

for (int j = 0; j < c; j++) {

tempMatrix[i][j] = matrix[i][j];

}

}

else if (i > row) {

for (int j = 0; j < c; j++) {

tempMatrix[i][j] = matrix[i-1][j];

}

}

}

// Chèn hàng mới vào vị trí row trong mảng mới

for (int j = 0; j < c; j++) {

tempMatrix[row][j] = rowArr[j];

}

for (int i = 0; i < r; i++) {

delete[] matrix[i];

}

delete[] matrix;

matrix = tempMatrix;

}