PRELAB

[Tiếng Việt]

Cho số nguyên n và số nguyên dương e, trong đó n là số cần tính lũy thừa và e là số mũ. Hãy viết một hàm đệ quy

```
int calculate_power(int n, int e){}
```

để tính giá trị của n^e.
Lưu ý không được sử dụng các từ khóa như for, while, goto (thậm chí là tên biến, comment).
Trong bài tập này đã khai báo #include <iostream> và using namespace std;
[English]

Given integer n and positive integer e, where n is the base and e is the exponent. Write a recursive function

```
int calculate_power(int n, int e){}
```

to calculate the value of n^e.
Please note that you can't using key work for, while, goto (even in variable names, comment).
For this exercise, we have #include <iostream> and using namespace std;

For example:

Test	Result

`int n = 2;`
`int e = 3;`
`cout << calculate_power(n ,e);`
	8

RECURSION 2

[Tiếng Việt]
Một hàm tìm ước số chung lớn nhất của 2 số nguyên dương có thể viết thông qua đệ quy và vòng lặp đơn giản. Bạn hãy viết hàm **gcdRecursion** để hiện thực tìm ước chung lớn nhất bằng đệ quy và
hàm **gcdIteration** để tìm ước số chung lớn nhất bằng vòng lặp
Đầu vào:
Lần lượt 2 số nguyên p, q ($1 \leq p,q < 10^9$).
Đầu ra:
Hàm **gcdRecursion** và **gcdIteration** lần lượt trả về giá trị là ước chung lớn nhất của p, q

[English]
A function that finds the greatest common divisor of two positive integers can be written through simple recursion and looping. You write the function gcdRecursion to perform the greatest common divisor by recursion and the function gcdIteration to find the greatest common divisor by loop.
Input:
Two integers p, q respectively ($1 \leq p,q < 109$).
Output:

The gcdRecursion and gcdIteration functions return the greatest common divisor of p, q, respectively.

Template of full code:

```
#include<iostream>
#include<string>
#include <string>
#include <sstream>
#include <fstream>
#include <vector>
using namespace std;

/* END of library */



int gcdRecursion(int p, int q)
{
    // BEGIN YOUR IMPLEMENTATION [1]
    // TODO

    // END YOUR IMPLEMENTATION [1]
}

int gcdIteration(int p, int q)
{
    // BEGIN YOUR IMPLEMENTATION [2]
    // TODO

    // END YOUR IMPLEMENTATION [2]
    return 0;
}
```

```
int main()
{
    hiddenCheck();
    int p,q;
    cin>>p>>q;
    cout<<gcdRecursion(p,q)<< " "<<gcdIteration(p,q);
    return 0;
}
```

For example:

Test	Input	Result
1	5 5	5 5

RECURSION 3

[Tiếng Việt]

Cho một chuỗi, hiện thực hàm

```
int strLen(char* str){}
```

để tính độ dài của chuỗi sử dụng đệ quy.
Lưu ý không được sử dụng các từ khóa như for, while, goto (thậm chí là tên biến, comment).
Trong bài tập này đã khai báo #include <iostream> và using namespace std;

[English]

Given a string, implement function

```
int strLen(char* str){}
```

to calculate length of the string using recursion.
Please note that you can't using key work for, while, goto (even in variable names, comment).
For this exercise, we have #include <iostream> and using namespace std;

For example:

Test	Result
`char str[] = "Truong DH Bach Khoa";` `cout << strLen(str);`	19

Hiện thực hàm sau:
int getValueAtIndex(int *ptr, int k);
Trả về giá trị của tại vị trí được chỉ định trong mảng qua con trỏ.
Trong đó:

ptr là con trỏ tới phần tử đầu tiên trong mảng.

k là vị trí cần truy xuất phần tử (giá trị này không vượt quá độ dài của mảng).

Implement the following function:
int getValueAtIndex(int *ptr, int k);
Return value at the position of the index number through pointer.
Where:

ptr is a pointer to the first element in the array.

k is the access position (this value does not exceed the length of the array).
For example:

Test	Result

`int arr[] = {1, 2, 3, 4, 5};`
`int k = 3;`
`cout << getValueAtIndex(arr, k);`
	4

Hiện thực hàm sau:
void swap(int *ptr2, int *ptr2);
Thực hiện hoán đổi giá tri tại vi trí của 2 con trỏ trỏ tới.
Trong đó:
ptr1, ptr2 lần lượt là là các con trỏ trỏ tới vi trí cần hoàn đổi.

Implement the following function:
void swap(int *ptr2, int *ptr2);
Swap the value at the position that 2 pointers point to.
Where:
ptr1, ptr2 respectively are pointers to swapping positions.

For example:

Test	Result

`int a = 5;`
`int b = 6;`
`swap(&a, &b);`
`cout << a << ' ' << b;`
	6 5

Answer:(penalty regime: 0 %)

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

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

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`

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

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

Mô tả tiếng Việt:

Kho lưu trữ của tổ chức SCP chứa hàng loạt các vật thể dị thường. Mỗi vật thể dị thường được lưu trữ dưới struct `SCP` với các thông tin như sau:

- `id`: kiểu int, là mã định danh (hay mã vật thể), phân biệt giữa các vật thể với nhau.
- `objClass`: kiểu int, là phân loại của vật thể đó.
- `speConProcedures`: kiểu string, mô tả quy trình quản thúc đặc biệt của vật thể đó.
- `description`: kiểu string, mô tả về các đặc điểm của vật thể đó.
- `addendums`: kiểu string*, là một tập hợp của các phụ lục đính kèm, mô tả các thông tin bổ sung cho vật thể đó.
- `numAddendums`: kiểu int, là số lượng phụ lục đính kèm.

Hiện thực struct `SCP` với các yêu cầu trên.
Ghi chú: (Các) thư viện `iostream` và `string` đã được khai báo, và `namespace std` đã được sử dụng.

English version:

SCP Foundation's classified archives consist of records of paranormal objects. The information of each object is stored using struct `SCP` with following requirements:

- `id`: integer, the identifier (or item number) of the object.
- `objClass`: integer, the object class.
- `speConProcedures`: string, specification of the object's special containment procedures.
- `description`: string, description of the object.
- `addendums`: string*, array of addendums describing additional information about the object.
- `numAddendums`: integer, the number of addendums attached.

Declare struct `SCP` with mentioned requirements.
Note: Libraries `iostream` and `string` have been imported, and `namespace std` has been used.

Mô tả tiếng Việt:

Cho struct `SCP` lưu trữ thông tin các vật thể dị thường được mô tả như sau:

```
struct SCP {
    int id;
    int objClass;
    string speConProcedures;
    string description;
    string* addendums;
    int numAddendums;
};
```

Hiện thực một hàm với prototype sau:

```
void addAddendum(SCP &obj, string addendum);
```

Hàm thực hiện bổ sung một phụ lục `addendum` vào cuối danh sách phụ lục (addendums) của `obj`.
Ghi chú: (Các) thư viện `iostream` và `string` đã được khai báo, và `namespace std` đã được sử dụng.

English version:

Struct `SCP` used to store information about paranormal objects is declared as below:

```
struct SCP {
    int id;
    int objClass;
    string speConProcedures;
    string description;
    string* addendums;
    int numAddendums;
};
```

Implement the function with the following prototype:
`void addAddendum(SCP &obj, string addendum);`
The function appends a new `addendum` into the array of addendums of the object `obj`.
Note: Libraries `iostream` and `string` have been imported, and `namespace std` has been used.
For example:

Test

```
string* addendums = new string[1];
addendums[0] = "Document #055-1: An Analysis of SCP-055\nThe author puts forward the hypothesis that SCP-055

SCP obj {55, 2, "Object is kept within a five (5) by five (5) by two point five (2.5) meter square room.", "A
1};

addAddendum(obj, "Document #055-2: Report of Dr. John Marachek\nSurvey team #19-055-127BXE was successfully a

cout << obj.addendums[1];

delete [] obj.addendums;
```

INLAB

[Tiếng Việt]

Hoàn thành hàm `recursiveSum` sử dụng đệ quy để tính và trả về tổng các phần tử trong một mảng
Đầu vào:

- `int arr[]`: Mảng chứa các số nguyên cần tính tổng.
- `int size`: Số lượng phần tử trong mảng.

Đầu ra: `int` - Kết quả là tổng các phần tử trong mảng cho trước.
Lưu ý: Sinh viên không được sử dụng các từ khoá sau, kể cả trong tên biến và comment: **include, for, while, goto**

[English]
Complete the function `recursiveSum` using recursion to calculate and return the sum of all elements in an array.
Inputs:

- `int arr[]`: An array containing integers to be summed.
- `int size`: The size of the array.

Output: `int` - the result is the sum of the elements in the given array.
Note: Students are not allowed to use the following keywords, including in variable names or comments: **include, for, while, goto**.

For example:

Test	Result

`int arr[] = {1,2,3,4,5};`
`int size = 5;`
`cout << recursiveSum(arr, size);`
	15

`int arr[] = {10, -10, 20, -20, 30, -30};`
`int size = sizeof(arr) / sizeof(int);`
`cout << recursiveSum(arr, size);`
	0

[Tiếng Việt]

Cho một số thập phân dương làm đầu vào, chúng ta cần triển khai hàm

```
long int decimalToBinary(int decimal_number){}
```

để chuyển đổi số thập phân dương đã cho thành số nhị phân tương đương.
Xin lưu ý rằng bạn không thể sử dụng từ khóa for, while, goto (ngay cả trong tên biến, comment).

Đối với bài tập này, chúng ta có #include <iostream> và sử dụng namespace std;

[English]
Given a positive decimal number as input, we need to implement function

```
long int decimalToBinary(int decimal_number){}
```

to convert the given positive decimal number into equivalent binary number.
Please note that you can't using key work for, while, goto (even in variable names, comment).
For this exercise, we have #include <iostream> and using namespace std;

For example:

Test	Result
`cout << decimalToBinary(20);`	10100

[Tiếng Việt]

Palindrome là một chuỗi (string) mà nếu đảo ngược thứ tự các ký tự trong chuỗi ta vẫn có được chuỗi cũ.
Ví dụ : "ABCBA", "RADAR", "otto", "i am ma i", "C".
Hàm **palindrome** kiểm tra một chuỗi có là Palindrome đã được cho phía dưới.
a/ Viết một chương trình C++ sử dụng vòng lặp while đọc vào nhiều chuỗi và gọi hàm trên để kiểm tra các chuỗi đọc vào có phải là palindrome hay không. Vòng lặp sẽ chấm dứt khi người sử dụng đọc vào một chuỗi bắt đầu bằng *.
b/ Viết hàm palindromeRecursion sử dụng kỹ thuật gọi đệ quy.để kiểm tra một chuỗi có phải là palindrome hay không.
Đầu vào:
Các chuỗi ký tự s có độ dài không quá 1000 ký tự
Đầu ra:

Mỗi dòng trả về giá trị của hàm palindrome và palindromRecursion (xem ví dụ để biết thêm chi tiết)

[English]

Palindrome is a string (string) that if you reverse the order of characters in the string, we still get the old string. For example: "ABCBA", "RADAR", "otto", "i am ma i", "C".

The **palindrome** function checks if a string is the given palindrome.

a/ Write a C++ program that uses a while loop to read multiple strings and call the above function to check whether the input strings are palindrome or not. The loop will terminate when the user reads into a string starting with *.

b/ Write a function palindromeRecursion using recursive calling technique.to check a string is palindrome or not.

Input:

Character strings s with a length of no more than 1000 characters

Output:

Each line returns the value of the palindrome and palindromeRecursion functions (see example for more details)

Template of full code:

```
#include<iostream>
#include<string>
using namespace std;
/* END of library */
bool palindrome(string strg);
bool palindromeRecursion(string s)
{
    // BEGIN YOUR IMPLEMENTATION [1]
    // TODO

    // END YOUR IMPLEMENTATION [1]
}
int main()
{
    hiddenCheck();
    // BEGIN YOUR IMPLEMENTATION [2]
    // TODO

    // END YOUR IMPLEMENTATION [2]
    return 0;
}
```

For example:

Test	Input	Result
1	abccba abc a *	true true false false true true

Hiện thực hàm sau:

int calcSum(int *ptr, int n);

Tính và trả về tổng của các phần tử trong mảng 1 chiều được cho bởi con trỏ.

Trong đó:

ptr là con trỏ tới phần tử đầu tiên trong mảng.

n là kích thước của mảng.

Lưu ý: Bạn cần phải dùng dereference operator (*) để lấy giá trị của các phần tử trong mảng. Không được dùng subscript operator ([]).

Implement the following function:

int calcSum(int *ptr, int n);

Calculate and return the sum of elements of a 1-dimension array given by a pointer.

Where:

ptr is a pointer to the first element in the array.

n is the size of the array.

Note: You need to use the dereference operator (*) to get the values of the elements in the array. The subscript operator ([]) cannot be used.

For example:

Test	Result
int arr[] = {1, 2, 3, 4, 5}; cout << calcSum(arr, sizeof(arr) / sizeof(arr[0]));	15
int arr[] = {0, -1, 5, 6, -5, 1, -9, -10, -6, 3}; cout << calcSum(arr, sizeof(arr) / sizeof(arr[0]));	-16

Hiện thực hàm sau:

void add(int *ptr, int n, int k);

Thực hiện thêm phần tử vào cuối của mảng 1 chiều được cho bởi con trỏ.

Trong đó:

ptr là con trỏ tới phần tử đầu tiên trong mảng.

n, k lần lượt là kích thước của mảng và phần tử cần được thêm vào.

Implement the following function:

void add(int *ptr, int n, int k);

Insert element to the end of the 1-dimension array given by a pointer.

Where:

ptr is a pointer to the first element in the array.

n, k respectively is the size of the array and the element that need to be added.

For example:

Test	Result

`int arr[100] = {1, 2, 3, 4, 5};`
`int n = 5;`
`int k = 7;`
`add(arr, n, k);`
`cout << arr[n];`
	`7`

`int arr[100] = {3, 9, 20, 6, 18, 0, 16, 8, 15, 14};`
`int n = 10;`
`int k = 0;`
`add(arr, n, k);`
`cout << arr[n];`
	`0`

`int arr[100] = {15, 8, -14, 13, -17, -12, 10, -15, -9, 4, 1, 0, 16, -11, -5, 19, 17, -13, -18, -4, 0};`
`int n = 22;`
`int k = -100;`
`add(arr, n, k);`
`cout << arr[n];`	

Mô tả tiếng Việt:

Hãy hiện thực hàm readArray() được khai báo như sau:
int** readArray()
Hàm này sẽ đọc dữ liệu cho một ma trận 2 chiều, mỗi chiều có 10 phần tử. Các phần tử của ma trận sẽ được nhập vào từ bàn phím (từ phần tử a[0][0] cho đến a[9][9]). Tuy nhiên nếu phần tử a[i][j] được nhập là 0 thì tất cả các phần tử còn lại trên hàng (a[i][k], j<k<10) đều được tự động gán là là 0, chương trình sẽ đọc tiếp phần tử a[i+1][0] từ bàn phím. Hàm readArray sẽ trả về một con trỏ tới mảng 2 chiều đã nhập này.
Đầu vào: Các phần tử có trong mảng 2 chiều, mỗi phần tử là một số nguyên dương có giá trị không vượt quá 1000.
Đầu ra: Con trỏ tới mảng 2 chiều vừa tạo
English version:
Implement the function readArray() that is declared as below syntax:

int** readArray()
The function reads a two-dimensional matrix each of which consists of 10 elements. These elements are entered from the keyboard (from a[0][0] to a[9][9]). If a[i][j] is assigned to 0, all remained element of the row (a[i][k], j<k<10) will automatically assigned to 0, and the function will continue to input the next-row element from the keyboard. Moreover, this function also returns a pointer which points to the two-dimensional matrix just entered.
Input: The positive integer matrix's elements which not surpass 1000.
Output: The pointer that points to the two-dimensional matrix just entered.

For example:

Test	Input	Result
1	`1 2 3 4 5 6 7 8 9 10` `0`	`1 2 3 4 5 6 7 8 9 10` `0 0 0 0 0 0 0 0 0 0`

Test	Input	Result
	1 0 2 0 3 0 4 5 0 6 7 0 8 0 9 0 10 11 12 13 14 0	1 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 3 0 0 0 0 0 0 0 0 0 4 5 0 0 0 0 0 0 0 0 6 7 0 0 0 0 0 0 0 0 8 0 0 0 0 0 0 0 0 0 9 0 0 0 0 0 0 0 0 0 10 11 12 13 14 0 0 0 0 0
2	0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Mô tả tiếng Việt:

Hiện thực hàm **void addElement(int*& arr, int n, int val, int index)** nhận vào một mảng động arr có chính xác n phần tử và tiến hành chèn giá trị val vào vị trí thứ index.

Đầu vào: Mảng một chiều arr có kích thước n, giá trị cần chèn val và vị trí cần chèn index.

Đầu ra: Mảng arr sau khi chèn.

Lưu ý: Việc chèn phần tử vào mảng động phải được thực hiện bằng cách giải phóng mảng cũ có n phần tử và cấp phát mảng mới có n+1 phần tử.

English version:
Implement the function **void addElement(int*& arr, int n, int val, int index)** that inputs a dynamic array, arr, consisting of exactly n elements and insert a value, val, into the a specific position, index.
Input: The n-size dynamic array needs to be inserted the value, val, into the specific position, index.
Output: The dynamic array after insert.
Note: Insertion of elements into a dynamic array must be executed by freeing the old array and allocating new memory for the new one.
For example:

Test	Input	Result
1	2 2 3 1 1	2 1 3
2	2 2 3 1 2	2 3 1

Mô tả tiếng Việt:

Hiện thực hàm **int* flatten(int** matrix, int r, int c)** trả về một mảng một chiều được "làm phẳng" từ mảng hai chiều có kích thước r x c (bằng cách nối các hàng của mảng hai chiều lại với nhau).
Đầu vào: Mảng hai chiều có kích thước r x c.
Đầu ra: Mảng một chiều sau khi được "làm phẳng" từ mảng hai chiều đầu vào.

English version:

Implement the function **int* flatten(int** matrix, int r, int c)** tht returns a one-dimensional array flatten from a two-dimensional matrix of size r x c (by concating all the matrix rows).
Input: The two-dimensional matrix of size r x c
Output: The one-dimensional array flatten from the previous two-dimensional matrix.
For example:

Test	Input	Result
1	2 3 1 2 3 4 5 6	1 2 3 4 5 6
2	2 3 1 2 3 4 0 0	1 2 3 4 0 0
3	3 3 1 2 3 4 5 6 2 9 -99	1 2 3 4 5 6 2 9 -99
4	3 4 1 2 3 4 4 5 6 0 -1 8 8 100	1 2 3 4 4 5 6 0 -1 8 8 100
5	4 4 1 2 4 4 4 5 3 0 2 5 1 6 7 7 8 4	1 2 4 4 4 5 3 0 2 5 1 6 7 7 8 4
1	4 1 1 4 2 3	1 4 2 3
7	1 4 1 2 4 4	1 2 4 4

Mô tả tiếng Việt:

Hiện thực hàm **char* concatStr(char* str1, char* str2)** trả về một chuỗi là kết quả sau khi nối 2 chuỗi str1 và str2 thành một chuỗi duy duy nhất.
Đầu vào: Hai chuỗi str1 và str2.
Đầu ra: Chuỗi được nối từ 2 chuỗi con str1 và str2.
Lưu ý: Không được phép sử dụng các hàm hỗ trợ của thư viện string và string.h cho bài tập này.

English version:

Implement the function **char* concatStr(char* str1, char* str2)** that return a string merged from two smaller string str1 and str2.
Input: Two string str1 and str2.

Output: The string merged from two smaller string str1 and str2.
Note: The string and string.h library are not allowed to use for this exercise.
For example:

Test	Result

`char s1[] = "Hello, ";`
`char s2[] = "how are you?";`
`char* s = concatStr(s1, s2);`
`cout << s;`
`delete[] s;`
	`Hello, how are you?`

`char s1[] = "Nice to ";`
`char s2[] = "meet you.";`
`char* s = concatStr(s1, s2);`
`cout << s;`
`delete[] s;`
	`Nice to meet you.`

`char s1[] = "Nice ";`
`char s2[] = "to meet ";`
`char s3[] = "you.";`
`char* temp = concatStr(s1, s2);`
`char* s = concatStr(temp, s3);`
`cout << s;`
`delete[] s; delete[] temp;`
	`Nice to meet you.`

`char s1[] = "Ho Chi Minh ";`
`char s2[] = "University ";`
`char s3[] = "of Technology.";`
`char* temp = concatStr(s1, s2);`
`char* s = concatStr(temp, s3);`
`cout << s;`
`delete[] s; delete[] temp;`
	`Ho Chi Minh University of Technology.`

`char s1[] = "This question ";`
`char s2[] = "is as easy as ";`
`char s3[] = "the other.";`
`char* temp = concatStr(s1, s2);`
`char* s = concatStr(temp, s3);`
`cout << s;`
`delete[] s; delete[] temp;`
	`This question is as easy as the other.`

`char s1[] = "That's ";`
`char s2[] = "a good idea.";`
`char* s = concatStr(s1, s2);`
`cout << s;`
`delete[] s;`
	`That's a good idea.`

`char s1[] = "123";`
`char s2[] = "456";`
`char* s = concatStr(s1, s2);`
`cout << s;`
`delete[] s;`
	`123456`

`char s1[] = "";`
`char s2[] = "CSE";`
`char* s = concatStr(s1, s2);`
`cout << s;`
`delete[] s;`
	`CSE`

`char s1[] = "";`
`char s2[] = "";`
`char* s = concatStr(s1, s2);`
`cout << s;`
`delete[] s;`	

Mô tả tiếng Việt:

Chuyển vị của một ma trận 2 chiều là một phần quan trọng trong việc tính toán trên ma trận nói riêng và đại số tuyến tính nói chung.
Gọi B là ma trận sau khi chuyển vị của ma trận A thì ma trận B có tính chất là b[i][j] = a[j][i].
Hãy viết hàm **int** transposeMatrix(int** matrix, int r, int c)** thực hiện phép chuyển vị trên ma trận đã được đề cập bên trên.
Đầu vào:

- Con trỏ tới mảng 2 chiều. Mỗi phần tử trong mảng 2 chiều có giá trị trong khoảng (-1000; 1000).
- Kích thước mảng 2 chiều là 1 cặp số dương r, c. Trong đó: r là số hàng của ma trận, c là số cột của ma trận. Giá trị n không vượt quá 1000.

Đầu ra: Con trỏ trỏ tới mảng hai chiều sau khi được chuyển vị. trong trường hợp ma trận đầu vào rỗng, trả về con trỏ null.

English version:
Transposition of a two-dimensional matrix is an important term for matrix calculations in particular and linear algebra in general.
A matrix B transposed from a matrix A that satisfied the following formula b[i][j] = a[j][i].
Implement the function **int** transposeMatrix(int** matrix, int r, int c)** that perform the transposition of the matrix mentioned above.
Input:

- The pointer that points to a two-dimensional matrix each of whose elements is in the range (-1000; 1000).
- The size of the matrix consists of the number of row r and the number of column n.

Output: The pointer that points to transposed two-dimensional matrix. If the input matrix is empty, return the null pointer.
For example:

Test	Input	Result
1	2 2 1 2 3 4	1 3 2 4
2	1 1 1	1
3	3 3 1 2 3 4 5 6 7 8 9	1 4 7 2 5 8 3 6 9
4	4 4 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16	1 5 9 13 2 6 10 14 3 7 11 15 4 8 12 16
5	2 2 10 12 14 16	10 14 12 16

Test	Input	Result
6	2 3 1 2 3 4 5 6	1 4 2 5 3 6
7	1 3 1 2 3	1 2 3
8	3 1 1 1 2	1 1 2
9	0 0	NULL
10	1 2 1 2	1 2

Mô tả tiếng Việt:

Cho struct `SCP` lưu trữ thông tin các vật thể dị thường được mô tả như sau:

```
struct SCP {
    int id;
    int objClass;
    string speConProcedures;
    string description;
    string* addendums;
    int numAddendums;
};
```

Hiện thực một hàm với prototype sau:

```
SCP createSCP(int id, int objClass, string speConProcedures, string description, string* addendums, int numAddendums);
```

Hàm có chức năng khởi tạo instance của struct SCP với các thông tin đã truyền vào, sau đó trả về instance đó.

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

English version:

Struct `SCP` used to store information about paranormal objects is declared as below:

```
struct SCP {
    int id;
    int objClass;
    string speConProcedures;
    string description;
    string* addendums;
    int numAddendums;
};
```

Implement the function with the following prototype:

`SCP createSCP(int id, int objClass, string speConProcedures, string description, string* addendums, int numAddendums);`

The function returns a new instance of struct `SCP` with the passed parameters.

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

For example:

Test

```
string* addendums = new string[2];
addendums[0] = "Document #055-1: An Analysis of SCP-055\nThe author puts forward the hypothesis that SCP-055 w
addendums[1] = "Document #055-2: Report of Dr. John Marachek\nSurvey team #19-055-127BXE was successfully able

SCP obj = createSCP(55, 2, "Object is kept within a five (5) by five (5) by two point five (2.5) meter square

for (int i = 0; i < 2; i++) cout << obj.addendums[i] << "\n";

delete [] addendums;
```

```
string* addendums = new string[2];
addendums[0] = "Document #055-1: An Analysis of SCP-055\nThe author puts forward the hypothesis that SCP-055 w
addendums[1] = "Document #055-2: Report of Dr. John Marachek\nSurvey team #19-055-127BXE was successfully able

SCP obj = createSCP(55, 2, "Object is kept within a five (5) by five (5) by two point five (2.5) meter square

cout << obj.id;

delete [] addendums;
```

```
string* addendums = new string[2];
addendums[0] = "Document #055-1: An Analysis of SCP-055\nThe author puts forward the hypothesis that SCP-055 w
addendums[1] = "Document #055-2: Report of Dr. John Marachek\nSurvey team #19-055-127BXE was successfully able

SCP obj = createSCP(55, 2, "Object is kept within a five (5) by five (5) by two point five (2.5) meter square

cout << obj.objClass;

delete [] addendums;
```

```
string* addendums = new string[2];
addendums[0] = "Document #055-1: An Analysis of SCP-055\nThe author puts forward the hypothesis that SCP-055 w
addendums[1] = "Document #055-2: Report of Dr. John Marachek\nSurvey team #19-055-127BXE was successfully able

SCP obj = createSCP(55, 2, "Object is kept within a five (5) by five (5) by two point five (2.5) meter square

cout << obj.speConProcedures;
```

```
string* addendums = new string[2];
addendums[0] = "Document #055-1: An Analysis of SCP-055\nThe author puts forward the hypothesis that SCP-055 w
addendums[1] = "Document #055-2: Report of Dr. John Marachek\nSurvey team #19-055-127BXE was successfully able

SCP obj = createSCP(55, 2, "Object is kept within a five (5) by five (5) by two point five (2.5) meter square

cout << obj.description;

delete [] addendums;
```

```
string* addendums = new string[2];
addendums[0] = "Document #055-1: An Analysis of SCP-055\nThe author puts forward the hypothesis that SCP-055 w
addendums[1] = "Document #055-2: Report of Dr. John Marachek\nSurvey team #19-055-127BXE was successfully able

SCP obj = createSCP(55, 2, "Object is kept within a five (5) by five (5) by two point five (2.5) meter square

for (int i = 0; i < obj.numAddendums; i++) cout << obj.addendums[i] << "\n";
```

Test

```
delete [] addendums;
```

```
string* addendums = new string[2];
addendums[0] = "Document #055-1: An Analysis of SCP-055\nThe author puts forward the hypothesis that SCP-055 w
addendums[1] = "Document #055-2: Report of Dr. John Marachek\nSurvey team #19-055-127BXE was successfully able

SCP obj = createSCP(55, 2, "Object is kept within a five (5) by five (5) by two point five (2.5) meter square

for (int i = 0; i < obj.numAddendums; i++) cout << obj.addendums[i] << "\n";

delete [] addendums;
```

```
string* addendums = new string[2];
addendums[0] = "Document #055-1: An Analysis of SCP-055\nThe author puts forward the hypothesis that SCP-055 w
addendums[1] = "Document #055-2: Report of Dr. John Marachek\nSurvey team #19-055-127BXE was successfully able

SCP obj = createSCP(55, 2, "Object is kept within a five (5) by five (5) by two point five (2.5) meter square

for (int i = 0; i < obj.numAddendums; i++) cout << obj.addendums[i] << "\n";

delete [] addendums;
```

```
string* addendums = new string[2];
addendums[0] = "Document #055-1: An Analysis of SCP-055\nThe author puts forward the hypothesis that SCP-055 w
addendums[1] = "Document #055-2: Report of Dr. John Marachek\nSurvey team #19-055-127BXE was successfully able

SCP obj = createSCP(55, 2, "Object is kept within a five (5) by five (5) by two point five (2.5) meter square

for (int i = 0; i < obj.numAddendums; i++) cout << obj.addendums[i] << "\n";

delete [] addendums;
```

```
string* addendums = new string[2];
addendums[0] = "Document #055-1: An Analysis of SCP-055\nThe author puts forward the hypothesis that SCP-055 w
addendums[1] = "Document #055-2: Report of Dr. John Marachek\nSurvey team #19-055-127BXE was successfully able

SCP obj = createSCP(55, 2, "Object is kept within a five (5) by five (5) by two point five (2.5) meter square

for (int i = 0; i < obj.numAddendums; i++) cout << obj.addendums[i] << "\n";

delete [] addendums;
```

Mô tả tiếng Việt:

Cho struct `SCP` lưu trữ thông tin các vật thể dị thường được mô tả như sau:

`struct SCP { int id; int objClass; string speConProcedures; string description; string* addendums; int numAddendums; };`

Hiện thực một hàm với prototype sau:

`string convertToString(SCP obj);`

Hàm trả về một string duy nhất, với format chuẩn được quy định bởi tổ chức:

```
Item #: SCP-<Mã định danh, thêm 0 vào trước nếu chưa đủ 3 chữ số>
Object Class: <Safe nếu objClass là 0, Euclid nếu là 1, Keter nếu là 2>
Special Containment Procedures: <Quy trình quản thúc đặc biệt>
Description: <Mô tả vật thể>
<Phụ lục 0>
<Phụ lục 1>
```

```
...
<Phụ lục n>
```
Tham khảo ví dụ để hiểu về format đầu ra.

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

English version:

Struct `SCP` used to store information about paranormal objects is declared as below:

```
struct SCP { int id; int objClass; string speConProcedures; string description; string* addendums; int numAddendums; };
```

Implement the function with the following prototype:

```
string convertToString(SCP obj);
```

The function returns a string with the following format:

```
Item #: SCP-<Item number, zero-left-padded until it has 3 or more digits.>
Object Class: <Safe if objClass is 0, Euclid if 1, Keter if 2>
Special Containment Procedures: <The specification of the object's special containment procedures.>
Description: <The object's description.>
<Addendum 0>
<Addendum 1>
...
<Addendum n>
```

Refer to the example to understand the format.

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

For example:

Test

```
string* addendums = new string[2];
addendums[0] = "Document #055-1: An Analysis of SCP-055\nThe author puts forward the hypothesis that SCP-055 w
addendums[1] = "Document #055-2: Report of Dr. John Marachek\nSurvey team #19-055-127BXE was successfully able

SCP obj {55, 2, "Object is kept within a five (5) by five (5) by two point five (2.5) meter square room.", "Al

cout << convertToString(obj);

delete [] addendums;
```

```
SCP obj {2, 1, "SCP-002 is to remain connected to a suitable power supply at all times.", "SCP-002 resembles a
cout << convertToString(obj);
```

```
SCP obj {500, 0, "SCP-500 must be stored in a cool and dry place away from bright light.", "SCP-500 is a small
cout << convertToString(obj);
```

```
SCP obj {1487, 1, "SCP-1487 is to be kept in Humanoid Containment Chamber #3821 at Site 23.", "SCP-1487 is a f
cout << convertToString(obj);
```

```
string* addendums = new string[1];
addendums[0] = "Site 81/715 Protocol: [ACCESS RESTRICTED]\nPlease Access Using 715/5 Authorization Code";

SCP obj {715, 0, "SCP-715 is to be contained at its point of origin within the ---- City Mall in ----, Ohio.",

cout << convertToString(obj);

delete [] addendums;
```

Test

string* addendums = new string[2];
addendums[0] = "Document #055-1: An Analysis of SCP-055\nThe author puts forward the hypothesis that SCP-055 w
addendums[1] = "Document #055-2: Report of Dr. John Marachek\nSurvey team #19-055-127BXE was successfully able
 	
SCP obj {55, 2, "Object is kept within a five (5) by five (5) by two point five (2.5) meter square room.", "Al
 	
cout << convertToString(obj);
 	
delete [] addendums;

SCP obj {2, 1, "SCP-002 is to remain connected to a suitable power supply at all times.", "SCP-002 resembles a cout << convertToString(obj);
SCP obj {500, 0, "SCP-500 must be stored in a cool and dry place away from bright light.", "SCP-500 is a small cout << convertToString(obj);
SCP obj {1487, 1, "SCP-1487 is to be kept in Humanoid Containment Chamber #3821 at Site 23.", "SCP-1487 is a f cout << convertToString(obj);

string* addendums = new string[1];
addendums[0] = "Site 81/715 Protocol: [ACCESS RESTRICTED]\nPlease Access Using 715/5 Authorization Code";
 	
SCP obj {715, 0, "SCP-715 is to be contained at its point of origin within the ---- City Mall in ----, Ohio.",
 	
cout << convertToString(obj);
 	
delete [] addendums;

Answer:(penalty regime: 0 %)

Mô tả tiếng Việt:

Cho struct SCP lưu trữ thông tin các vật thể dị thường được mô tả như sau:

```
struct SCP {
    int id;
    int objClass;
    string speConProcedures;
    string description;
    string* addendums;
    int numAddendums;
};
```

Một đối tượng dị thường là SCP-038 có khả năng tạo bản sao của các vật thể dị thường khác, ví dụ như SCP-500 (thuốc vạn ứng). Khi đối tượng gốc bị tiêu thụ/phá hủy, bản sao vẫn tồn tại độc lập.
Hiện thực một hàm với prototype sau:

```
SCP* cloneSCP(SCP* original);
```

Hàm có chức năng tạo một bản sao con trỏ loại SCP chứa các thông tin như đối tượng gốc original, sau đó trả về con trỏ đó. Lưu ý về shallow copy và deep copy.
Ghi chú: (Các) thư viện `iostream` và `string` đã được khai báo, và `namespace std` đã được sử dụng.

English version:

Struct `SCP` used to store information about paranormal objects is declared as below:

```
struct SCP {
    int id;
    int objClass;
    string speConProcedures;
    string description;
    string* addendums;
    int numAddendums;
};
```

The paranormal object SCP-038 has the ability to clone other objects, such as SCP-500 (panacea). The clone is independent of the original, and the clone will remain the same whether the original is destroyed or modified.

Implement the function with the following prototype:

```
SCP* cloneSCP(SCP* original);
```

The function returns a pointer of the new cloned instance of the original object `original`. Choose the appropriate copy method.

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

For example:

Test

`string* addendums = new string[2];`
`addendums[0] = "Document #055-1: An Analysis of SCP-055\nThe author puts forward the hypothesis that SCP-055 w`
`addendums[1] = "Document #055-2: Report of Dr. John Marachek\nSurvey team #19-055-127BXE was successfully able`
 	
`SCP* obj = new SCP {55, 2, "Object is kept within a five (5) by five (5) by two point five (2.5) meter square`
 	
`SCP* newObj = cloneSCP(obj);`
 	
`obj->id = 1;`
`cout << obj->id << " " << newObj->id << "\n";`
 	
`delete [] obj->addendums;`
`delete obj;`
 	
`delete [] newObj->addendums;`
`delete newObj;`

`string* addendums = new string[2];`
`addendums[0] = "Document #055-1: An Analysis of SCP-055\nThe author puts forward the hypothesis that SCP-055 w`
`addendums[1] = "Document #055-2: Report of Dr. John Marachek\nSurvey team #19-055-127BXE was successfully able`
 	
`SCP* obj = new SCP {55, 2, "Object is kept within a five (5) by five (5) by two point five (2.5) meter square`
 	
`SCP* newObj = cloneSCP(obj);`
 	
`obj->description = "The object cannot be described.";`

Test

```
cout << obj->description << " " << newObj->description << "\n";

delete [] obj->addendums;
delete obj;

delete [] newObj->addendums;
delete newObj;
```

```
string* addendums = new string[2];
addendums[0] = "Document #055-1: An Analysis of SCP-055\nThe author puts forward the hypothesis that SCP-055 w
addendums[1] = "Document #055-2: Report of Dr. John Marachek\nSurvey team #19-055-127BXE was successfully able

SCP* obj = new SCP {55, 2, "Object is kept within a five (5) by five (5) by two point five (2.5) meter square

SCP* newObj = cloneSCP(obj);

obj->addendums[0] = "Access authority required.";
cout << obj->addendums[0] << " " << newObj->addendums[0] << "\n";

delete [] obj->addendums;
delete obj;

delete [] newObj->addendums;
delete newObj;
```

```
string* addendums = new string[2];
addendums[0] = "Document #055-1: An Analysis of SCP-055\nThe author puts forward the hypothesis that SCP-055 w
addendums[1] = "Document #055-2: Report of Dr. John Marachek\nSurvey team #19-055-127BXE was successfully able

SCP* obj = new SCP {55, 2, "Object is kept within a five (5) by five (5) by two point five (2.5) meter square

SCP* newObj = cloneSCP(obj);

delete [] obj->addendums;
delete obj;

cout << newObj->id << "\n";

delete [] newObj->addendums;
delete newObj;
```

```
{
string* addendums = new string[2];
addendums[0] = "Document #055-1: An Analysis of SCP-055\nThe author puts forward the hypothesis that SCP-055 w
addendums[1] = "Document #055-2: Report of Dr. John Marachek\nSurvey team #19-055-127BXE was successfully able

SCP* obj = new SCP {55, 2, "Object is kept within a five (5) by five (5) by two point five (2.5) meter square

SCP* newObj = cloneSCP(obj);

delete [] obj->addendums;
delete obj;

cout << newObj->addendums[0] << "\n";

delete [] newObj->addendums;
delete newObj;
}
```

```
string* addendums = new string[2];
```

Test

```
addendums[0] = "Document #055-1: An Analysis of SCP-055\nThe author puts forward the hypothesis that SCP-055 w
addendums[1] = "Document #055-2: Report of Dr. John Marachek\nSurvey team #19-055-127BXE was successfully able

SCP* obj = new SCP {55, 2, "Object is kept within a five (5) by five (5) by two point five (2.5) meter square

SCP* newObj = cloneSCP(obj);

obj->id = 1;
cout << obj->id << " " << newObj->id << "\n";

delete [] obj->addendums;
delete obj;

delete [] newObj->addendums;
delete newObj;
```

```
string* addendums = new string[2];
addendums[0] = "Document #055-1: An Analysis of SCP-055\nThe author puts forward the hypothesis that SCP-055 w
addendums[1] = "Document #055-2: Report of Dr. John Marachek\nSurvey team #19-055-127BXE was successfully able

SCP* obj = new SCP {55, 2, "Object is kept within a five (5) by five (5) by two point five (2.5) meter square

SCP* newObj = cloneSCP(obj);

obj->description = "The object cannot be described.";
cout << obj->description << " " << newObj->description << "\n";

delete [] obj->addendums;
delete obj;

delete [] newObj->addendums;
delete newObj;
```

```
string* addendums = new string[2];
addendums[0] = "Document #055-1: An Analysis of SCP-055\nThe author puts forward the hypothesis that SCP-055 w
addendums[1] = "Document #055-2: Report of Dr. John Marachek\nSurvey team #19-055-127BXE was successfully able

SCP* obj = new SCP {55, 2, "Object is kept within a five (5) by five (5) by two point five (2.5) meter square

SCP* newObj = cloneSCP(obj);

obj->addendums[0] = "Access authority required.";
cout << obj->addendums[0] << " " << newObj->addendums[0] << "\n";

delete [] obj->addendums;
delete obj;

delete [] newObj->addendums;
delete newObj;
```

```
string* addendums = new string[2];
addendums[0] = "Document #055-1: An Analysis of SCP-055\nThe author puts forward the hypothesis that SCP-055 w
addendums[1] = "Document #055-2: Report of Dr. John Marachek\nSurvey team #19-055-127BXE was successfully able

SCP* obj = new SCP {55, 2, "Object is kept within a five (5) by five (5) by two point five (2.5) meter square

SCP* newObj = cloneSCP(obj);

delete [] obj->addendums;
delete obj;
```

Test

```
cout << newObj->id << "\n";

delete [] newObj->addendums;
delete newObj;
```

```
{
string* addendums = new string[2];
addendums[0] = "Document #055-1: An Analysis of SCP-055\nThe author puts forward the hypothesis that SCP-055 w
addendums[1] = "Document #055-2: Report of Dr. John Marachek\nSurvey team #19-055-127BXE was successfully able

SCP* obj = new SCP {55, 2, "Object is kept within a five (5) by five (5) by two point five (2.5) meter square

SCP* newObj = cloneSCP(obj);

delete [] obj->addendums;
delete obj;

cout << newObj->addendums[0] << "\n";

delete [] newObj->addendums;
delete newObj;
}
```

POSTLAB

[Tiếng Việt]

Hiện thực hàm `findOccurrences` với ba đối số: một mảng số nguyên `nums`, độ dài của mảng `size` và một số nguyên `element`. Hàm sẽ trả về số lần xuất hiện của `element` trong `nums`.

Ví dụ:

```
Input: nums = {1,2,3}, size = 3, element = 3

Output: 1
```

Lưu ý: Xin lưu ý rằng bạn không thể sử dụng từ khóa for, while, goto (ngay cả trong tên biến, comment). Bạn có thể triển khai các hàm đệ quy khác nếu cần.
Đối với bài tập này, chúng ta có #include <iostream> và using namespace std;

[English]

Implement the function `findOccurrences` with three parameters: an array of integers `nums`, the length of the array `size`, and an integer `element`. The function will return the number of occurrences of the `element` in `nums`.

Example:

```
Input: nums = {1,2,3}, size = 3, element = 3

Output: 1
```

Note: Please note that you cannot use the keywords for, while, goto (even in variable names, comments). You may implement additional recursive functions if necessary.
For this exercise, we have #include <iostream> and using namespace std;

For example:

Test	Result
`int nums[] = {1,2,3};` `cout << findOccurrences(nums, 3, 3);`	1

[Tiếng Việt]

Cho một số nguyên x, thực hiện hàm

```
int countWaySumOfSquare(int x)
```

để tìm số cách biểu thị x thành tổng bình phương của **các số nguyên dương duy nhất**.
Ví dụ:

```
Input  : x = 100
Output : 3
Giải thích: 100 = 10^2 = 8^2 + 6^2 = 1^2 + 3^2 + 4^2 + 5^2 + 7^2
```

Lưu ý: Xin lưu ý rằng bạn không thể sử dụng từ khóa for, while, goto (ngay cả trong tên biến, comment). Bạn có thể triển khai các hàm đệ quy khác nếu cần.

Đối với bài tập này, chúng ta có #include <iostream>, #include <math.h> và using namespace std;

[English]
Give an integer x, implement function

```
int countWaySumOfSquare(int x)
```

to find number of ways to express x as sum of squares of **unique positive integers**.
For example:

```
Input  : x = 100
Output : 3
Explain: 100 = 10^2 = 8^2 + 6^2 = 1^2 + 3^2 + 4^2 + 5^2 + 7^2
```

Note: Please note that you can't using key work for, while, goto (even in variable names, comment). You can implement other recursive functions if needed.
For this exercise, we have #include <iostream>, #include <math.h> and using namespace std;

For example:

Test	Result
`cout << countWaySumOfSquare(100);`	3

Hiện thực hàm sau:
int findMax(int *ptr, int n);
Tìm và trả về phần tử lớn nhất trong mảng 1 chiều được cho bởi con trỏ.
Trong đó:
ptr là con trỏ tới phần tử đầu tiên trong mảng.
n là kích thước của mảng.

Implement the following function:
int findMax(int *ptr, int n);
Find and return the maximum element of a 1-dimension array given by a pointer.
Where:
ptr is a pointer to the first element in the array.
n is the size of the array.

For example:

Test	Result
`int arr[] = {1, 2, 3, 4, 5};` `cout << findMax(arr, sizeof(arr) / sizeof(arr[0]));`	5

Hiện thực hàm sau:
void reverse(int *ptr, int n);
Đảo ngược mảng 1 chiều được cho bởi con trỏ.
Trong đó:
ptr là con trỏ tới phần tử đầu tiên trong mảng.
n là kích thước của mảng 1 chiều.

Lưu ý: Bạn cần phải dùng dereference operator (*) để lấy giá trị của các phần tử trong mảng. Không được dùng subscript operator ([]).

Implement the following function:
void findMax(int *ptr, int n);
Reverse the 1-dimension array given by a pointer.
Where:
ptr is a pointer to the first element in the array.
n is the size of the array.
Note: You need to use the dereference operator (*) to get the values of the elements in the array. The subscript operator ([]) cannot be used.

For example:

Test	Result
`int arr[] = {1, 2, 3, 4, 5};` `reverse(arr, sizeof(arr) / sizeof(arr[0]));`	`5, 4, 3, 2, 1`

Test	Result

cout << arr[0];
for (int i = 1; i < sizeof(arr) / sizeof(arr[0]); i++)
{
    cout << ", " << arr[i];
}	

Hiện thực hàm sau:

bool isSymmetry(int *head, int *tail);

Kiểm tra mảng 1 chiều có phải là một mảng đối xứng hay không.

Trong đó:

head, tail lần lượt là con trỏ tới phần tử đầu tiên và cuối cùng trong mảng.

Implement the following function:

bool isSymmetry(int *head, int *tail);

Checks if the 1-dimensional array is a symmetric array.

Where:

head, tail respectively are pointers to the first element and last element of the array.

Lưu ý: Sinh viên chỉ có 5 lần nộp không tính penalty, ở lần nộp thứ 6 trở đi bài làm sẽ được tính là 0 điểm.

For example:

Test	Result
int arr[] = {1, 2, 1}; cout << isSymmetry(arr, arr + (sizeof(arr) / sizeof(arr[0])) - 1);	1

Mô tả tiếng Việt:

Cho một mảng động hai chiều matrix có kích thước r x c. Hiện thực hàm **int** insertCol(int**& matrix, int r, int c, int* colArr, int col)** tiến hành chèn mảng rowArr (có kích thước r) vào cột thứ col của matrix.
Đầu vào: Mảng 2 chiều matrix có kích thước r x c, cột cần chèn colArr và vị trí chèn col.
Đầu ra: Mảng 2 chiều matrix sau khi được chèn, đầu ra phải được điều chỉnh trên biến matrix truyền vào.

English version:

Given a dynamic two-dimensional array of size r x c. Implement the function **int** insertCol(int**& matrix, int r, int c, int* colArr, int col)** that can insert the colArr array (with the size r) into the column position, col, of the matrix.
Input: The two-dimensional matrix of size r x c, the insert column rowArr and the insert position col.
Output: The two-dimensional matrix after insert.

For example:

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

Test	Input	Result
	4 5 6 2 7 8	

Mô tả tiếng Việt:

Cho một mảng động hai chiều matrix có kích thước r x c. Hiện thực hàm **bool deleteRow(int**& matrix, int r, int c, int row)** tiến hành xóa hàng thứ row của matrix.

Đầu vào: Mảng 2 chiều matrix có kích thước r x c và vị trí hàng cần xóa row.
Đầu ra: Mảng 2 chiều matrix sau khi xóa hàng và kết quả xóa thành công hay không.
Lưu ý: Nếu mảng hai chiều xóa đi hàng cuối cùng, mà trận được truyền vào hàm sẽ trả về giá trị nullptr và đồng thời giải phóng toán bộ vùng nhớ của mảng hai chiều được cấp phát trước đó.
English version:
Given a dynamic two-dimensional array of size r x c. Implement the function **bool deleteRow(int**& matrix, int r, int c, int row)** that can remove a row from the matrix.
Input: The two-dimensional matrix of size r x c and the remove position row.
Output: The two-dimensional matrix after removing the row.
Note: If the final row of the matrix is removed, the matrix parameter will return nullptr value as well as deallocate the memory previously used by the matrix.

For example:

Test	Input	Result
1	2 3 1 2 3 4 5 6 0	4 5 6

Mô tả tiếng Việt:

Cho một mảng động hai chiều matrix có kích thước r x c. Hiện thực hàm **bool deleteCol(int** matrix, int r, int c, int col)** tiến hành xóa cột thứ col của matrix.
Đầu vào: Mảng 2 chiều matrix có kích thước r x c và vị trí cột cần xóa **col**.
Đầu ra: Mảng 2 chiều matrix sau khi xóa cột.
Lưu ý: Nếu mảng hai chiều xóa đi cột cuối cùng, kết quả trả về là nullptr và đồng thời giải phóng toán bộ vùng nhớ của mảng hai chiều được cấp phát trước đó.
English version:

Given a dynamic two-dimensional array of size r x c. Implement the function **bool deleteCol(int**& matrix, int r, int c, int col)** that can remove a column from the matrix.
Input: The two-dimensional matrix of size r x c and the remove position col.
Output: The two-dimensional matrix after removing the column.
Note: If the final column of the matrix is removed, the function will return nullptr value as well as deallocate the memory previously used by the matrix.

For example:

Test	Input	Result
1	2 3 1 2 3 4 5 6 2	1 2 4 5

Mô tả tiếng Việt:

Cho struct `SCP` lưu trữ thông tin các vật thể dị thường được mô tả như sau:

```
struct SCP {
    int id;
    int objClass;
    string speConProcedures;
    string description;
    string* addendums;
    int numAddendums;
};
```

Chỉ số `objClass` của mỗi vật thể dị thường đánh giá mức độ khó khăn trong việc quản thúc đối tượng. Chỉ số này càng cao, đối tượng càng khó bị quản thúc, và việc quản thúc cần tiêu tốn nhiều tài nguyên hơn.
Hiện thực một hàm với prototype sau:

```
int compareObjectClass(const SCP &objA, const SCP &objB);
```

Hàm so sánh mức độ khó khăn trong việc quản thúc của hai đối tượng `objA` và `objB` và trả về `-1` nếu `objA` dễ quản thúc hơn `objB`, `1` nếu `objA` khó quản thúc hơn `objB`, `0` nếu mức độ khó khăn trong việc quản thúc của hai vật thể này tương đương nhau.
Ghi chú: (Các) thư viện `iostream` và `string` đã được khai báo, và `namespace std` đã được sử dụng.

English version:

Struct `SCP` used to store information about paranormal objects is declared as below:

```
struct SCP {
    int id;
    int objClass;
    string speConProcedures;
    string description;
    string* addendums;
```

```
    int numAddendums;
};
```

The `objClass` of each object indicates how difficult an object is to contain. The higher the value, the more resources and budget are needed to contain the object.

`int compareObjectClass(const SCP &objA, const SCP &objB);`

The function returns `-1` if `objA` is easier to be contained than `objB`, `1` if `objA` is more difficult to be contained than `objB`, or `0` if the difficulties in containing two objects are equivalent.

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

For example:

Test

`SCP objA {2, 1, "SCP-002 is to remain connected to a suitable power supply at all times.", "SCP-002 resembles`
`SCP objB {55, 2, "Object is kept within a five (5) by five (5) by two point five (2.5) meter square room.", "A`
`cout << compareObjectClass(objA, objB);`

Mô tả tiếng Việt:

Cho struct `SCP` lưu trữ thông tin các vật thể dị thường được mô tả như sau:

```
struct SCP {
    int id;
    int objClass;
    string speConProcedures;
    string description;
    string* addendums;
    int numAddendums;
};
```

Tổ chức SCP lưu trữ thông tin các vật thể dị thường dưới dạng một array được cấp phát động. Tuy nhiên, vì một vài lý do, các hồ sơ không được sắp xếp theo một trật tự nào.

Hiện thực một hàm với prototype sau:

`SCP** sortDatabase(SCP** arr, int n);`

Trong đó `arr` là một array chứa các pointer trỏ đến các instance kiểu SCP, `n` là độ dài của mảng. Hàm thực hiện sắp xếp theo chiều tăng dần mã định danh các đối tượng SCP và trả về mảng sau khi sắp xếp.

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

English version:

Struct `SCP` used to store information about paranormal objects is declared as below:

```
struct SCP {
    int id;
    int objClass;
    string speConProcedures;
    string description;
    string* addendums;
    int numAddendums;
};
```

SCP Foundation's classified archives are stored as a dynamic allocated array. However, because of some specific reason, the array is not sorted in any order.
Implement the function with the following prototype:

```
SCP** sortDatabase(SCP** arr, int n);
```

Where `arr` is the array of `SCP` typed pointers, `n` is the length of the array. The function returns the sorted array in item number (`id`) increasing order.
Note: Libraries `iostream` and `string` have been imported, and `namespace std` has been used.
For example:

Test

```
int n = 3;
SCP** data = new SCP* [n];
data[0] = new SCP {2, 1, "SCP-002 is to remain connected to a suitable power supply at all times.", "SCP-002 r
data[1] = new SCP {55, 2, "Object is kept within a five (5) by five (5) by two point five (2.5) meter square r
data[2] = new SCP {49, 1, "SCP-049 is contained within a Standard Secure Humanoid Containment Cell in Research

data = sortDatabase(data, n);

for (int i = 0; i < n; i++) cout << data[i]->id << " ";

for (int i = 0; i < n; i++) {
    delete [] data[i]->addendums;
    delete data[i];
}
delete [] data;
```