Hiện thực hàm bằng ngôn ngữ C/C++ void replaceString(string s, string s1, string s2) để in ra chuỗi s sau khi đã thay thế  chuỗi con s1 cuối cùng thành s2 có trong s.

Nếu không tìm thấy chuỗi s1 thì giữ nguyên chuỗi s và in ra.

**Gợi ý: Tham khảo hàm string::replace trong thư viện <string>**

Implement the function by C/C++: void replaceString(string s, string s1, string s2) to print the string s after replacing the last substring s1 with the s2 contained in s. If the string s1 is not found, keep the string s and print it.

**Hint:** Refer to the string::replace function in the <string> library.

#include <iostream>

#include <string>

using namespace std;

void replaceString(string s, string s1, string s2) {

size\_t pos = s.rfind(s1); // tìm vị trí cuối cùng của chuỗi s1 trong chuỗi s

if (pos != string::npos) { // nếu tìm thấy chuỗi s1

s.replace(pos, s1.length(), s2); // thay thế chuỗi s1 bằng chuỗi s2

}

cout << s << endl; // in ra chuỗi kết quả

}

int main() {

string s = "Hello world! This is a test string.";

string s1 = "string";

string s2 = "sentence";

replaceString(s, s1, s2);

return 0;

Hiện thực hàm bằng C/C++ deleteWord(string s, string s1) với chức năng in ra chuỗi s sau khi xóa tất cả các chuỗi s1 có trong s.

#include <iostream>

#include <string>

using namespace std;

void deleteWord(string s, string s1) {

size\_t pos = s.find(s1); // tìm vị trí đầu tiên của chuỗi s1 trong chuỗi s

while (pos != string::npos) { // nếu tìm thấy chuỗi s1

s.erase(pos, s1.length()); // xóa chuỗi s1

pos = s.find(s1, pos); // tìm vị trí tiếp theo của chuỗi s1 trong chuỗi s

}

cout << s << endl; // in ra chuỗi kết quả

}

int main() {

string s = "Hello world! This is a test string.";

string s1 = "is";

deleteWord(s, s1);

return 0;

}

Viết chương trinh bằng C/C++ đọc vào chuỗi **s**, tìm tất cả các chuỗi con trong **s** mà chuỗi con chỉ có duy nhất 1 kí tự (ví dụ : aaaa, bbbb, cc, ...) và trả về độ dài của chuỗi con dài nhất.

**Đầu vào:**

Chuỗi s có độ dài ngắn nhất là 1.

**Đầu ra:**

Độ dài của chuỗi con dài nhất mà chỉ có sự xuất hiện của 1 ký tự trong chuỗi con.

#include <iostream>

#include <string>

using namespace std;

int main() {

string s;

cin >> s; // đọc chuỗi s từ input

int maxLength = 1; // độ dài của chuỗi con dài nhất

int currentLength = 1; // độ dài của chuỗi con hiện tại

char currentChar = s[0]; // ký tự hiện tại

for (int i = 1; i < s.length(); i++) { // duyệt từ ký tự thứ 2 đến hết chuỗi

if (s[i] == currentChar) { // nếu ký tự hiện tại xuất hiện trong chuỗi con

currentLength++; // tăng độ dài chuỗi con hiện tại lên 1

} else { // nếu ký tự hiện tại không xuất hiện trong chuỗi con

maxLength = max(maxLength, currentLength); // cập nhật độ dài của chuỗi con dài nhất

currentLength = 1; // reset độ dài của chuỗi con hiện tại về 1

currentChar = s[i]; // gán ký tự hiện tại bằng ký tự mới

}

}

maxLength = max(maxLength,

Write a program by C/C++ that reads into a string **s**, finds all substrings in **s** whose substring has only 1 character (for example: aaaa, bbbb, cc, ...) and returns the length of the longest substring.  
**Input:**

A string s with at least 1 character.

**Output:**

the length of sub-string that only has 1 kind of character

Trong chương trình này, đầu tiên chúng ta đọc vào chuỗi s từ bàn phím. Sau đó, chúng ta duyệt qua các ký tự trong chuỗi s bằng cách sử dụng vòng lặp for. Trong quá trình duyệt, chúng ta sử dụng hai biến để theo dõi ký tự hiện tại và độ dài của chuỗi con hiện tại. Nếu ký tự hiện tại trùng với ký tự tiếp theo, chúng ta tăng độ dài của chuỗi con hiện tại. Nếu ký tự hiện tại khác với ký tự tiếp theo, chúng ta xét độ dài của chuỗi con hiện tại và cập nhật độ dài của chuỗi con dài nhất n

#include <iostream>

#include <string>

using namespace std;

/\*Viết chương trình đọc vào chuỗi s, tìm tất cả các chuỗi con trong s mà chuỗi con chỉ có duy nhất 1 kí tự (ví dụ : aaaa, bbbb, cc, ...) và trả về độ dài của chuỗi con dài nhất.

Đầu vào:

Chuỗi s có độ dài ngắn nhất là 1.

Đầu ra:

Độ dài của chuỗi con dài nhất mà chỉ có sự xuất hiện của 1 ký tự trong chuỗi con.\*/

int main() {

string s;

cin >> s; // đọc vào chuỗi s

int max\_length = 0;

char current\_char = s[0];

int current\_length = 1;

for (int i = 1; i < s.length(); i++) {

if (s[i] == current\_char) { // ký tự hiện tại trùng với ký tự tiếp theo

current\_length++;

}

else {

if (current\_length == 1) { // độ dài của chuỗi con hiện tại là 1

max\_length = max(max\_length, 1); // cập nhật độ dài

}

else { // độ dài của chuỗi con hiện tại lớn hơn 1

max\_length = max(max\_length, current\_length); // cập nhật độ dài

}

current\_char = s[i]; // chuyển sang ký tự tiếp theo

current\_length = 1;

}

}

// Cập nhật vị trí trùng cuối

if (current\_length == 1) {

max\_length = max(max\_length, 1); // cập nhật độ dài của chuỗi con dài nhất

}

else {

max\_length = max(max\_length, current\_length); // cập nhật độ dài của chuỗi con dài nhất

}

cout << max\_length << endl;

return 0;

}

Cho mảng 2 chiều chứa các số nguyên, kích thước M x N. Hiện thực hàm bằng C++: int ascendingRows(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 hàng trong mảng được gọi là **HN1** nếu trong hàng đó, mỗi phần tử đều có giá trị không lớn hơn các phần tử đứng sau nó. Tìm số hàng **HN1** có trong mảng.

**Mô tả tiếng Việt:**

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

Hiện thực hàm:

int primeColumns(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 cột của mảng được gọi là **HN2** nếu tổng tất cả các phần tử trong cột đó là số nguyên tố. Tìm số cột **HN2** có trong mảng.

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 integer, its size is M x N.

Implement the following function:

int primeColumns(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 column of the given array is called as **HN2** if the sum of all elements on it is a prime number. Find the number of **HN2** columns in 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] = {{-64,-28,-3,64},{-56,90,57,-31}};  cout << primeColumns(arr,2,4); | 0 |
| int arr[][1000] = {{34,-15,11,-70,-23,24},{-39,-90,63,-45,-52,48},{-42,92,55,92,82,81}};  cout << primeColumns(arr,3,6); | 1 |

Answer:(penalty regime: 0 %)

#include <iostream>

using namespace std;

int ascendingRows(int arr[][1000], int row, int col) {

int count = 0;

for (int i = 0; i < row; i++) { // duyệt qua tất cả các hàng trong mảng

bool isAscending = true; // biến đánh dấu xem hàng hiện tại có đang tăng dần hay không

for (int j = 0; j < col - 1; j++) { // duyệt qua tất cả các cột trừ cột cuối cùng trong hàng hiện tại

if (arr[i][j] > arr[i][j+1]) { // nếu phần tử hiện tại lớn hơn phần tử kế tiếp

isAscending = false; // đánh dấu hàng không đang tăng dần

break; // thoát khỏi vòng lặp

}

}

if (isAscending) { // nếu hàng đang tăng dần

count++; // tăng biến đếm số hàng tăng dần lên 1

}

}

return count; // trả về số hàng tăng dần trong mảng

}

int main() {

int arr[3][1000] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};

int row = 3;

int col = 3;

int count = ascendingRows(arr, row, col);

cout << "Number of ascending rows: " << count << endl;

return 0;

}

Trong ví dụ này, chúng ta duyệt qua tất cả các hàng trong mảng bằng cách sử dụng hai vòng lặp for lồng nhau. Trong mỗi hàng, chúng ta kiểm tra xem các phần tử có được sắp xếp theo thứ tự tăng dần hay không bằng cách so sánh mỗi phần tử với phần tử kế tiếp của nó. Nếu phần tử hiện tại lớn hơn phần tử kế tiếp, chúng ta đánh dấu hàng hiện tại không đang tăng dần và thoát khỏi vòng lặp. Nếu tất cả các phần tử trong hàng đều được sắp xếp theo thứ tự tăng dần, chúng ta tăng biến đếm số hàng tăng dần lên 1. Cuối cùng, chúng ta trả về số hàng tăng dần trong mảng.

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

Hiện thực hàm bằng C++: int primeColumns(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 cột của mảng được gọi là **HN2** nếu tổng tất cả các phần tử trong cột đó là số nguyên tố. Tìm số cột **HN2** có trong mảng.

Hiện thực hàm:

int primeColumns(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 cột của mảng được gọi là **HN2** nếu tổng tất cả các phần tử trong cột đó là số nguyên tố. Tìm số cột **HN2** có trong mảng.

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

int primeColumns(int arr[][1000], int row, int col)

{

int count = 0;

for(int j=0; j<col; j++) // duyệt qua từng cột

{

int sum = 0;

bool isPrime = true;

for(int i=0; i<row; i++) // tính tổng các phần tử trong cột

{

sum += arr[i][j];

}

if(sum > 0) // nếu tổng khác 0 mới kiểm tra số nguyên tố

{

for(int k=2; k<=sqrt(sum); k++)

{

if(sum%k == 0)

{

isPrime = false;

break;

}

}

if(isPrime) // nếu tổng là số nguyên tố thì tăng biến đếm

{

count++;

}

}

}

return count;

}

Giải thích:

* Duyệt qua từng cột của mảng.
* Tính tổng các phần tử trong cột đó.
* Kiểm tra nếu tổng khác 0 mới kiểm tra xem có phải số nguyên tố hay không.
* Tăng biến đếm nếu cột đó có tổng là số nguyên tố.
* Trả về số cột HN2 tìm được.

Viết hàm bằng C++: **void uppercase(string output)** để đọc vào chuỗi S từ bàn phím, sau đó chuyển tất các ký tự trong chuỗi S thành ký tự viết HOA và xuất kết quả ra file output. Chú ý: chỉ thay đổi các chữ cái in thường, các kí tự khác sẽ được giữ nguyên.

**Đầu vào:**

Biến "output" chứa tên file dùng để xuất kết quả.

**Đầu ra:**

Hàm đọc chuỗi S từ bàn phím và xử lý chuỗi như mô tả. Sau đó ghi chuỗi đã xử lý vào file có tên được chứa trong biến “output” (hàm không trả về kết quả).

Viết hàm bằng C++: **void threeChars(string fileName)** đọc vào các hàng từ một file có đuôi txt, mỗi hàng chứa một chuỗi có 3 kí tự. Xác định xem 3 ký tự trên mỗi hàng có đúng thứ tự bảng chữ cái (ASCII) hay không, nếu đúng xuất ra "true", ngược lại xuất ra "false". Chương trình sẽ lặp cho đến khi 3 ký tự đọc vào là "\*\*\*".

**Đầu vào:**

Biến fileName là tên file chứa dữ liệu cần đọc vào.

**Đầu ra:**

"true" hoặc "false", theo sau bởi dấu xuống dòng, trừ trường hợp đọc vào "\*\*\*" thì kết thúc chương trình.

Viết hàm **void process(string fileName)** đọc một file đuôi txt, dòng đầu gồm 2 số N và M cách nhau bởi 1 khoảng trắng.

N dòng theo sau, mỗi dòng gồm M số thực cách nhau bởi 1 khoảng trắng.

Trả về giá trị lớn nhất của mỗi dòng và giá trị lớn nhất trong tất cả các số.  
Các thư viện đã được include: iostream, fstream, string.  
  
**Đầu vào:**

Biến "fileName" là tên file chứa dữ liệu đầu vào.

**Đầu ra:**

Giá trị lớn nhất của mỗi dòng và giá trị lớn nhất của tất cả các số, cách nhau bởi ký tự khoảng trắng.