# PRELAB

## C-string

### [Q 1.1] Loại bỏ các khoảng trắng thừa

Viết hàm void process(const char \*str, char \*outstr) loại bỏ các khoảng trắng thừa trong chuỗi sao cho không còn hai khoảng trắng liền kề nhau và không có khoảng trắng nào ở đầu và cuối chuỗi.

- Đầu vào:

Con trỏ char \* đến kí tự đầu tiên của chuỗi

- Đầu ra:

Kết quả của bài toán được ghi vào chuỗi outstr

- Các thư viện có thể dùng: <iostream>, <cstring>

- **Note**: Sinh viên không được sử dụng các từ khoá sau trong đoạn code nộp bài, kể cả trong comment: "include", "string"

---------------------------------------

Implement void process(const char \*str, char \*outstr) function which eliminates redundant space characters so that there are no consecutive space characters and there is no space character at the beginning/end of string.

- Input:

Char \* pointer to the first character of string

- Output:

The result is written to outstr

- Available library: <iostream>, <cstring>

- **Note**: Your code MUST NOT contain any of the words in the list {"include", "string"} even in the comment

For example:

| **Test** | **Result** |
| --- | --- |
| char str[] = " abc def ghi " ;  char \*outstr = new char[strlen(str) + 1];  process(str, outstr);  cout << outstr; | abc def ghi |

// Student may implement another function as need

void process(char str[], char outstr[]) {

// TODO

}

### Answer q1.1 from quang vinh

void process(char str[], char outstr[])

{

// TODO

int b=0;

int a= strlen(str);

for(int k=0;k<a;k++)

{

if(str[k]!= ' ')

{

for(int i=k;i< a;i++)

{

if((str[i] == ' ' && str[i+1] != ' ' ) || (str[i] != ' ') )

{

outstr[b]= str[i];

b++;

}

}

break;

}

}

outstr[b]='\0';

}

### [Q 1.2] Đảo ngược các ký tự trong chuỗi

Viết hàm void process(const char \*str, char \*outstr) đảo ngược các ký tự trong chuỗi.

- Đầu vào:

Con trỏ char \* đến ký tự đầu tiên của chuỗi

- Đầu ra:

Kết quả của bài toán được ghi vào chuỗi outstr

- Các thư viện có thể dùng: <iostream>, <cstring>

- **Note**: Sinh viên không được sử dụng các từ khoá sau trong đoạn code nộp bài, kể cả trong comment: "include", "string"

---------------------------------------

Implement void process(const char \*str, char \*outstr) function which reverses the string.

- Input:

Char \* pointer to the first character of string

- Output:

The result is written to outstr

- Available library: <iostream>, <cstring>

- **Note**: Your code MUST NOT contain any of the words in the list {"include", "string"} even in the comment

For example:

| **Test** | **Result** |
| --- | --- |
| char str[] = "this is my school" ;  char \*outstr = new char[strlen(str) + 1];  process(str, outstr);  cout << outstr; | loohcs ym si siht |

// Student may implement another function as need

void process(const char \*str, char \*outstr) {

// TODO

}

### Answer q1.2 from le minh

// Student may implement another function as need

void process(const char \*str, char \*outstr) {

// TODO

int length = strlen(str);

for(int i=1; i<= length; i++)

{

outstr[i-1] = str[length-i];

}

outstr[length]=’\0’;

//**attention: function don’t auto set position of \0, the error will be random alphabet, number.**

}

### [Q 1.5] Tính giá trị của một chuỗi bit nhị phân

Viết hàm int calc(char \*str) trả về giá trị dạng thập phân của chuỗi các bit nhị phân

- Đầu vào:

Con trỏ char \* đến phần tử đầu tiên của chuỗi

- Đầu ra:

Giá trị dạng thập phân của chuỗi các bit nhị phân

- Các thư viện có thể dùng: <iostream>, <cstring>

- **Note**: Sinh viên không được sử dụng các từ khoá sau trong đoạn code nộp bài, kể cả trong comment: "include", "string"

---------------------------------------

Implement int calc(const char \*str) function which returns the decimal value of binary string str.

- Input:

Char \* pointer to the first character of string

- Output:

The decimal value of binary string

- Available library: <iostream>, <cstring>

- **Note**: Your code MUST NOT contain any of the words in the list {"include", "string"} even in the comment

For example:

| **Test** | **Result** |
| --- | --- |
| char str[] = "001011" ;  cout << calc(str); | 11 |

// Student may implement another function as need

int calc(char str[]) {

// TODO

}

### Answer q1.5 from le minh

// Student may implement another function as need

int twopowern(int n)

{

if(n==0) return 1;

else

{

return 2\*twopowern(n-1);

}

}

int calc(char str[])

{

// TODO

int length = strlen(str);

int result = 0;

for(int i=0; i<length; i++)

{

if(str[i] == '0')

{

continue;

}

else

{

result = result + twopowern(length-i-1);

}

}

return result;

}

## Class string

### [Q 2.7] In ra một phần của chuỗi

**[Tiếng Việt]**

Hiện thực hàm cutString(string s, int index) để in ra chuỗi con của chuỗi s từ vị trí index đến hết (index tính từ 0).

**Gợi ý :**Sử dụng hàm string::substr trong thư viện <string>.

**[English ]**

Implement the function cutString(string s, int index) to print the substring of string s from index position to the end (index from 0).

**Hint:**  Use the string::substr function in the <string> library.

For example:

| **Test** | **Result** |
| --- | --- |
| string s = "Truong Dai Hoc Bach Khoa.";  cutString(s, 7); | Dai Hoc Bach Khoa. |

void cutString(string s, int index){

// TO DO

}

### Answer 2.7 from le minh

void cutString(string s, int index)

{

// TO DO

if(index >= s.length())

**//avoid error out of range**

{

cout<<"";

}

else

{

string result = s.substr(index);

cout<<result;

}

}

### [Q 2.6] Tìm tất cả vị trí xuất hiện của chuỗi con trong chuỗi s

**[Tiếng Việt ]**

Hiện thực hàm findAllIndex(string s1, string s2) để in ra tất cả các vị trí xuất hiện của kí tự đầu tiên của chuỗi s2 trong chuỗi s1. Nếu không tìm thấy in ra -1. Các vị trí tìm thấy sẽ cách nhau một khoảng trắng, sau vị trí cuối cùng, không in thêm bất kỳ ký tự nào (kể cả khoảng trắng, dấu xuống hàng)

**Gợi ý:** Sử dụng hàm string::find trong thư viện <string>.

**[English ]**

Implement findAllIndex(string s1, string s2) to print all positions of the first character of string s2 in string s1. If not found, print -1. The found positions will be separated by a space, after the last position, do not print any more characters (including spaces, carriage returns)

**Hint:**  Use the string::find function in the <string> library.

For example:

| **Test** | **Result** |
| --- | --- |
| string s1 = "Truong Dai Hoc Bach Khoa.";  string s2 = "a";  findAllIndex(s1, s2); | 8 16 23 |

void findAllIndex(string s1, string s2){

// TO DO

}

### Answer 2.6 from le minh

void findAllIndex(string s1, string s2)

{

// TO DO

int found0 = s1.find(s2,0);

**//RETURN VALUE OF FIND FUNCTION**

**//The position of the first character of the first match.**

**//If no matches were found, the function returns string::npos.**

int found1;

if(found0 == string::npos)

{

cout<<-1;

return;

}

//loop i=0 to i<= s1.length()

for(int i=1; i<=s1.length(); i++)

{

found1 = s1.find(s2,found0+1);

if(found1 == string::npos)

{

cout<<found0;

break;

}

else

{

cout<<found0<<' ';

}

found0=found1;

}

}

## **Multi-dimensional Array**

### [Q 3.1] Find Column with Max Sum

#### dung câu hỏi

**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 |

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

}

### Answer q3.1 from le minh (đọc đề kĩ vào – findmaxcolumn hay findmaxrow)

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

{

int sum\_row[col];

//get the sum of every row

for(int i=0; i<col; i++)

{

for(int j=0; j<row; j++)

{

if(j==0)

{

sum\_row[i]=arr[0][i];

continue;

}

sum\_row[i]=sum\_row[i]+arr[j][i];

}

}

//get the index

int max=sum\_row[0];

int index=0;

for(int i=1; i<col; i++)

{

if(sum\_row[i]>=max)

{

max=sum\_row[i];

index=i;

}

}

return index;

}

### [Q 3.2] Diagonal Product

**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 |

Bottom of Form

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

}

### Answer q3.2 from leminh

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

{

int result;

for(int i=0; i<row; i++)

{

if(i==0)

{

result=arr[i][i];

continue;

}

result=result\*arr[i][i];

}

return result;

}

### [Q 3.3] Symmetric Matrix

**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 |

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

}

### Answer q3.3 from leminh

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

{

for(int i=0; i<row; i++)

{

for(int j=0; j<row; j++)

{

if(arr[i][j] != arr[j][i]) return false;

}

}

return true;

}

### [Q 3.7] Diagonals Difference

**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 |

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

}

### Answer q3.7 from leminh

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

{

int N=row; //row=col

//diagonal: duong cheo

//diagonal1: left to right

int LR=arr[x][y];

//toLEFT

for(int i=x-1,j=y-1; i>=0 && j>=0; i--,j--)

{

LR=LR+ arr[i][j];

}

//toRIGHT

for(int i=x+1,j=y+1; i<N && j<N; i++, j++)

{

LR=LR+ arr[i][j];

}

//diagonal2: right to left

int RL=arr[x][y];

//toLEFT

for(int i=x+1,j=y-1; i<N && j>=0; i++, j--)

{

RL=RL+ arr[i][j];

}

//toRIGHT

for(int i=x-1,j=y+1; i>=0 && j<N; i--, j++)

{

RL=RL+ arr[i][j];

}

if(LR-RL >=0) return LR-RL;

else return RL-LR;

}

## File IO

### [Q 4.2] - calculate sum

Viết hàm **void calSum(string fileName)** để tính tổng của các số nguyên không âm được đọc vào từ một file đuôi txt (các số nằm trên 1 hàng phân cách với nhau bằng một khoảng trắng).

**Đầu vào:**

Biến "fileName" là tên file dữ liệu chứa các số nguyên không âm (các số nằm trên 1 hàng phân cách với nhau bằng một khoảng trắng).

**Đầu ra:**

Một số nguyên là tổng của các số nguyên đầu vào.

Write a function **void calSum(string fileName)** to calculate the sum of non-negative integers read from a txt file (numbers in a row separated by a space).

**Input:**

The variable "fileName" is a data file name containing non-negative integers (numbers in 1 row separated by a space).

**Output:**

An integer is the sum of the input integers.

For example:

| **Test** | **Input** | **Result** |
| --- | --- | --- |
| 1 | 0 2 3 5 | 10 |

### Answer q4.2 from leminh 0.9/1.0 don’t know testcase error

void calSum(string fileName)

{

// TODO

//open file

fstream afile;

afile.open(fileName,ios::in );

if(!afile)

{

return;

}

//input integers

int N=100;

int array[N];

int n=0;

int result = 0;

while(!afile.eof())

**//afile.eof() is Boolean value**

{

afile>>array[n];

cin.ignore();

result= result + array[n];

n++;

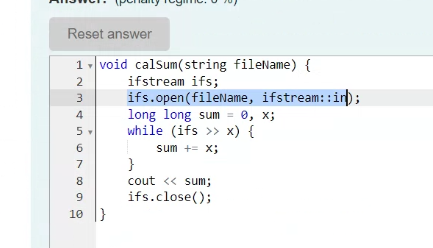
}

afile.close();

//print

cout<<result;

}



# INLAB

## C-string

### [Q 1.3] Normalize people's names

Viết hàm void process(const char \*str, char \*outstr) chuẩn hóa tên người (tên tiếng Anh). Quy tắc chuẩn hóa như sau:

1. Chữ cái đầu tiên của mỗi từ phải ghi hoa, các chữ cái còn lại ghi thường.

2. Tên người chỉ bao gồm những ký tự trong bảng chữ cái alphabet tiếng Anh.

3. Không có các khoảng trắng thừa. Cụ thể, không có 2 khoảng trắng liền kề nhau và không có khoảng trắng nào ở đầu và cuối chuỗi.

- Đầu vào:

Con trỏ char \* đến kí tự đầu tiên của chuỗi

- Đầu ra:

Tên người đã được chuẩn hóa được ghi vào chuỗi outstr

- Các thư viện có thể dùng: <iostream>, <cstring>

- **Note**: Sinh viên không được sử dụng các từ khoá sau trong đoạn code nộp bài, kể cả trong comment: "include", "string"

---------------------------------------

Implement void process(const char \*str, char \*outstr) function which standardizes English name. The standardized name has the following features:

1. The first character of each word must be uppercase. The remaining characters of each word must be lowercase.

2. Only contains characters in English alphabet.

3. There are no consecutive space characters and there is no space character at the beginning/end of the name

- Input:

Char \* pointer to the first character of string

- Output:

The standardized name is written to outstr

- Available library: <iostream>, <cstring>

- **Note**: Your code MUST NOT contain any of the words in the list {"include", "string"} even in the comment

For example:

| **Test** | **Result** |
| --- | --- |
| char str[] = "doAn VAN \*(()hau @!#$" ;  char \*outstr = new char[strlen(str) + 1];  process(str, outstr);  cout << outstr; | Doan Van Hau |

// Student may implement another function as need

void process(const char \*name, char \*outstr)

{

// TODO

}

### [Q 1.4] Find the first word in a string that is repeated

Viết hàm void printFirstRepeatedWord(const char str[]) in ra từ đầu tiên trong chuỗi bị lặp lại. 

- Đầu vào:

Con trỏ char \* đến phần tử đầu tiên của chuỗi

- Đầu ra:

In ra từ đầu tiên trong chuỗi có lặp lại. Nếu không có từ nào lặp lại thì in ra "No Repetition"

- Các thư viện có thể dùng: <iostream>, <cstring>

- **Note**: Sinh viên không được sử dụng các từ khoá sau trong đoạn code nộp bài, kể cả trong comment: "include", "string"

---------------------------------------

Implement void printFirstRepeatedWord(const char str[]) function which print the first word that repeats.

- Input:

Char \* pointer to the first character of string

- Output:

Print the first word that has repetition. If there is no repeated word, print "No Repetition"

- Available library: <iostream>, <cstring>

- **Note**: Your code MUST NOT contain any of the words in the list {"include", "string"} even in the comment

For example:

| **Test** | **Result** |
| --- | --- |
| char str[] = "car taxi bike bike car taxi" ;  printFirstRepeatedWord(str); | car |

// Student may implement another function as need

void printFirstRepeatedWord(char str[]) {

// TODO

}

P

## Class string

### [Q 2.4] Replace last occurrence of substring s1 with substring s2

**[Tiếng Việt]**

Hiện thực hàm 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>.**

**[English]**

Implement the function 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.

For example:

| **Test** | **Result** |
| --- | --- |
| string s = "Truong Dai Hoc Bach Khoa, Sai Gon";  string s1 = "ai";  string s2 = "ong";  replaceString(s, s1, s2); | Truong Dai Hoc Bach Khoa, Song |

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

//TO DO

}

### Answer q2.4

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

{

//TO DO

int found = s.find(s1);

if (found != std::string::npos)

{

//have s1 in s

int prefound;

while(found != std::string::npos)

{

prefound = found;

found= s.find(s1,found+1);

**//tìm vị trí có string s1 từ vị trí found+1 trong string s.**

}

s.replace(prefound,s1.length(),s2);

**//thay thế s1.length() kí tự trong string s kể từ vị trí prefound**

}

cout<<s;

}

### [Q 2.5] Delete a word from the string.iệt ]

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

**[English ]**

Implement deleteWord(string s, string s1) function with function to print string s after deleting all strings s1 contained in s.

For example:

| **Test** | **Result** |
| --- | --- |
| string s = "Truong Dai Hoc Bach Khoa, Sai Gon";  string s1 = "ai ";  deleteWord(s, s1); | Truong DHoc Bach Khoa, SGon |

void deleteWord(string s, string s1){

//TO DO

}

### Answer 2.5

void deleteWord(string s, string s1)

{

//TO DO

int found = s.find(s1);

while(found != std::string::npos)

{

s.erase(found,s1.length());

//**xóa s1.length() kí tự từ vị trí found của string s**

found=found-s1.length();

found = s.find(s1,found+1);

}

cout<<s;

}

### [Q 2.1] Find the maximum length of a substring made from only one character

**[Tiếng Việt]**

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.

**[English ]**

Write a program 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.

For example:

| **Test** | **Input** | **Result** |
| --- | --- | --- |
| 1 | abcabbbcbb | 3 |

#include <iostream>

#include <string>

using namespace std;

int main() {

// TODO

}

### Answer q2.1

#include <iostream>

#include <string>

using namespace std;

int len(string s, int index)

{

if(index==s.length()-1) return 1;

**//== ko phải =, cái sai hết sức tào lao**

if(s[index] != s[index+1]) return 1;

else

{

return 1 + len(s,index+1);

}

}

int main()

{

// TODO

string s;

cin>>s;

int max=1;

int trying;

for(int i=0; i<s.length(); i++)

{

trying = len(s,i);

if(trying>=max) max=trying;

}

cout<<max;

return 0;

}

## **Multi-dimensional Array**

### [Q 3.4] Ascending Rows

**Mô tả tiếng Việt:**

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

Hiện thực hàm:

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.

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 M x N.

Implement the following function:

int ascendingRows(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 row of the given array is called as **HN1** if on this row, each element's value is not higher than all elements after it. Find the number of **HN1** rows of the given array.

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

For example:

| **Test** | **Result** |
| --- | --- |
| int arr[][1000] = {{32,4,9},{-80,37,71},{-91,-79,-55}};  cout << ascendingRows(arr,3, 3); | 2 |
| int arr[][1000] = {{-28,-8,-60,18},{-100,44,-1,24},{-94,92,-70,75}};  cout << ascendingRows(arr,3,4); | 0 |

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

}

### Answer q3.4

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

{

int result=0;

for(int i=0; i<row; i++)

{

for(int j=0; j<col-1; j++)

{

if(arr[i][j] <= arr[i][j+1])

{

if(j==col-2) result++;

continue;

}

else break;

}

}

return result;

}

### [Q 3.5] Prime Columns

**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 |

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

}

### Answer q3.5

bool isPrime(int n)

{

// Corner case

if (n <= 1)

return false;

// Check from 2 to n-1

for (int i = 2; i < n; i++)

if (n % i == 0)

return false;

return true;

}

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

{

int result =0;

int sum[col]={};

for(int i=0; i<col; i++)

{

for(int j=0; j<row; j++)

{

sum[i]= sum[i] + arr[j][i];

}

if(isPrime(sum[i]) == true) result++;

}

return result;

}

## File IO

### [Q 4.3] uppercase

Viết hàm **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).

Write a function **void uppercase(string output)** to read the string S from the keyboard, then convert all characters in string S to uppercase characters and output the result to the output file.

Note: change only lowercase letters, other characters will not change.

**Input:**

The variable "output" contains the filename used to output the result.

**Output:**

The function reads the string S from the keyboard and processes the string as described. Then write the processed string to a file whose name is contained in the variable “output” (the function does not return results).

For example:

| **Test** | **Input** | **Result** |
| --- | --- | --- |
| 1 | heLlO10 | HELLO10 |

void uppercase(string output) {

// TODO

}

### Answer q4.3

void uppercase(string output)

{

// TODO

string s;

cin>>s;

for(int i=0; i<s.length(); i++)

{

if(s[i] >=97 && s[i] <=122) s[i]=s[i] -32;

**//ASCII: a-z: 97-122, A-Z 65-90**

}

fstream afile;

afile.open(output,ios::out);

afile<<s;

afile.close();

}

### [Q 4.1] 3 chars in order

Viết hàm **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.

Write a function **void threeChars(string fileName)** that reads in rows from a txt file, each row containing a string of 3 characters. Determines if 3 characters per row are in correct alphabetical order (ASCII), if true output "true", otherwise "false". The program will loop until 3 characters read in is "\*\*\*".

**Input:**

The variable "fileName" is the name of the file containing the input data.

**Output:**

"true" or "false", followed by a newline, unless reading "\*\*\*" ends the program.

For example:

| **Test** | **Input** | **Result** |
| --- | --- | --- |
| 1 | 123  abc  aab  \*\*\* | true  true  true |

void threeChars(string fileName) {

// TODO

}

### Answer q4.1

void threeChars(string fileName)

{

// TODO

fstream afile;

afile.open(fileName,ios::in);

string s;

afile>>s;

while(s != "\*\*\*")

{

for(int i=0; i<s.length()-1; i++)

{

if(s[i] > s[i+1])

{

cout<<"false"<<endl;

break;

}

else

{

if(i==s.length()-2) cout<<"true"<<endl;

else continue;

}

}

afile>>s;

}

}

### [Q 4.4] find numbers

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.

Write a function **void process(string fileName)** that reads a txt file, the first line consists of 2 numbers N and M separated by a space.

N lines follow, each containing M real numbers separated by 1 space.

Print the maximum value of each line and the maximum value of all numbers.

Included libraries: iostream, fstream, string.

**Input:**

The variable "fileName" is the name of the file containing the input data.  
**Output:**

The maximum value of each line and the maximum value of all numbers, separated by a space character.

For example:

| **Test** | **Input** | **Result** |
| --- | --- | --- |
| 1 | 3 4  1 2 3 4  4.3 41.2 -4.576 0  -3 -2 -4 -1 | 4 41.2 -1 41.2 |

void process(string fileName) {

// TODO

}

### Answer q4.4

void process(string fileName)

{

// TODO

fstream afile;

afile.open(fileName,ios::in);

int N,M;

afile>>N;

afile>>M;

double array[N][M];

for(int i=0; i<N; i++)

{

for(int j=0; j<M; j++)

{

afile>>array[i][j];

}

}

double max=array[0][0];

double MAX[N];

for(int i=0; i<N; i++)

{

MAX[i]=array[i][0];

}

for(int i=0; i<N; i++)

{

for(int j=0; j<M; j++)

{

if(array[i][j] >= MAX[i]) MAX[i]=array[i][j];

if(array[i][j] >= max) max = array[i][j];

}

cout<<MAX[i]<<' ';

}

cout<<max;

}

# POSTLAB

## C-string

## Class string

### [Q 2.3] Tìm độ dài lớn nhất của chuỗi con có tính chất đối xứng

**[Tiếng Việt]**

Viết chương trình đọc vào chuỗi **s** từ bàn phím, tìm và trả về chuỗi con có tính chất đối xứng dài nhất có trong **s**.  
  
**Đầu vào:**

Chuỗi s.

**Đầu ra:**

Chuỗi con có tính chất đối xứng dài nhất có trong **s**. Trong trường hợp có nhiều chuỗi con dài nhất có cùng độ dài, trả về chuỗi con có vị trí bắt đầu nhỏ nhất.

**[English ]**

Write a program read an **s** string from keyboard, find and return the longest symetrical sub-string of **s**.  
  
**Input:**

String s.

**Output:**

The longest symmetrical sub-string from **s**. In case where there are more than one highest string, return the string with the smallest starting character index.

For example:

| **Test** | **Input** | **Result** |
| --- | --- | --- |
| 1 | babad | bab |

#include <iostream>

#include <string>

using namespace std;

int main() {

// TODO

}

### [Q 2.2] Tìm độ dài lớn nhất của chuỗi con không có ký tự nào lặp lại

**[Tiếng Việt ]**

Viết chương trình đọc vào chuỗi **s** từ bàn phím, tìm và trả về độ dài của chuỗi con dài nhất không có ký tự nào được lặp lại có trong **s**.  
  
**Đầu vào:**

Chuỗi s.

**Đầu ra:**

Độ dài của chuỗi con dài nhất không có ký tự nào được lặp lại có trong **s**.

**[English ]**

Write a program read a string **s** from keyboard, find and return the length of the longest sub-string that doesn't have any repeated character in **s**.  
  
**Input:**

String s.

**Output:**

Length of the longest sub-string that doesn't have any repeated character in **s**.

For example:

| **Test** | **Input** | **Result** |
| --- | --- | --- |
| 1 | abcabcbb | 3 |

## **Multi-dimensional Array**

### [Q 3.6] Special Cells

**Mô tả tiếng Việt:**

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

Hiện thực hàm:

int specialCells(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 ô trong mảng được gọi là **HN3** nếu tổng tất cả các phần tử trong hàng chứa ô đó và tổng tất cả các phần tử trong cột chứa ô đó đều là số nguyên tố. Tìm số ô **HN3** 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 specialCells(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 cell is called as **HN3** if the sum of all elements of the row containing it and the sum of all elements of the column containing it are prime numbers. Find the number of **HN3** cells 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] = {{37,53,74},{12,37,60},{98,13,15}};  cout << specialCells(arr,3,3); | 2 |
| int arr[][1000] = {{1,87,26},{97,95,88},{57,60,46}};;  cout << specialCells(arr,3,3); | 0 |

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

}

### [Q 3.8] Odd Sub Matrixes

**Mô tả tiếng Việt:**

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

Hiện thực hàm:

int subMatrix(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 mảng con kích thước 2x2 thuộc mảng đã cho được gọi là **HN4** nếu tổng tất cả các phần tử trong nó là một số lẻ. Tìm số mảng con **HN4** trong mảng đã cho.

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 subMatrix(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 sub-array whose size is 2x2 is called as **HN4** if the sum of all elements of it is an odd number. Find the number of sub-array 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] = {{66,16,71},{25,81,61},{2,10,34}};  cout << subMatrix(arr,3, 3); | 1 |
| int arr[][1000] ={{44,45,89},{82,91,34},{83,87,33},{65,51,66}};  cout << subMatrix(arr,4, 3); | 4 |

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

}

## File IO

### [Q 4.5] student grading

Viết hàm **void studendGrading(string fileName)** đọc một file đuôi txt, dòng đầu gồm 1 số nguyên dương N, N dòng theo sau mỗi dòng chứa 4 số thực lần lượt là điểm số của các môn NMĐT,  KTLT, DSA và PPL của N học sinh.

Điểm trung bình (ĐTB) của sinh viên sẽ là trung bình cộng của 4 cột điểm trên. Sinh viên sẽ được xếp loại dựa trên ĐTB như sau:

* Loại A nếu ĐTB >= 8 và không có môn nào dưới 5.
* Loại B nếu 8 > ĐTB >= 6.5 và không có môn nào dưới 5.
* Loại C nếu 6.5 > ĐTB >= 5 và không có môn nào dưới 5
* Loại D cho các trường hợp còn lại

Xác định số lượng sinh viên mỗi loại và xuất kết quả ra màn hình.

**Đầu vào:**

Biến "fileName" là tên file chứa chứa thông tin về điểm số của sinh viên.

**Đầu ra:**

Số lượng sinh viên mỗi loại được ghi ra màn hình.

Write a function **void studendGrading(string fileName)** that reads a txt file, the first line consists of a positive integer N, N lines follow each line containing 4 real numbers, respectively, the scores of the subjects NMDT, KTLT, DSA and PPL of N students.

The student's grade point average (GPA) will be the average of the above 4 score columns. Students will be graded based on the following scores:

* Grade A if the test score is >= 8 and no subject is below 5.
* Grade B if 8 > GPA >= 6.5 and no subject below 5.
* Grade C if 6.5 > GPA >= 5 and no subject below 5.
* Grade D for the remaining cases.

Determine the number of students of each category and output the results to the screen.

**Input:**

The variable "fileName" is the file name that contains information about the student's score.

**Output:**

The number of students in each category is recorded on the screen.

For example:

| **Test** | **Input** | **Result** |
| --- | --- | --- |
| 1 | 2  8 8 8 6  9 9 9 5 | A 1  B 1  C 0  D 0 |

void studentGrading(string fileName) {

// TODO

}

### [Q 4.6] calculate money

Viết hàm **void calMoney(string price, string buy)** đọc vào 2 file price.txt và buy.txt.

Trong đó:

* File price.txt chứa thông tin về các sản phẩm, gồm: Dòng đầu tiên chứa số nguyên dương N (N <= 20) là số lượng sản phẩm hiện có trong cửa hàng. N dòng tiếp theo mỗi dòng chứa 2 số nguyên dương là ID và giá của các sản phẩm tương ứng

     Ví dụ:

          3

          1 10

          2 15

          3 12

Có nghĩa là: hiện tại có 3 sản phẩm được bán trong cửa hàng, sản phẩm 1 giá 10 sản phẩm 2 giá 15 và sản phẩm 3 có giá là 12.

* File buy.txt chứa thông tin về việc mua hàng hóa của khách hàng. Dòng đầu tiên chứa số nguyên dương M là số lượng khách hàng mua hàng trong 1 ngày. M dòng tiếp theo mỗi dòng chứa thông tin như sau: sẽ có  giá nhiều trị nằm trên một hàng, cách nhau bởi một khoảng trắng. Giá trị đầu tiên sẽ là tên khách hàng (dạng string), các giá trị còn lại sẽ đi theo cặp với nhau, tương ứng là: <ID sản phẩm>\_<số lượng tương ứng>

     Ví dụ:

          2

          A 1 2 2 3

          B 1 3 3 2

Có nghĩa là có 2 khách hàng A và B, A mua 2 sản phẩm loại 1 và 3 sản phẩm loại 2; B mua 3 sản phẩm loại 1 và 2 sản phẩm loại 3.

Lưu ý: ID của sản phẩm và tên khách hàng là duy nhất (không lặp lại).

Tính số tiền mỗi khách hàng cần trả, sau đó xuất kết quả ra màn hình.

**Đầu vào:**

2 biến price (là tên của file chứa thông tin sản phẩm) và buy (là tên của file chứa thông tin mua hàng).

**Đầu ra:**

Số tiền mỗi khách hàng cần trả.

Write a function **void calMoney(string price, string buy)** that reads into 2 files price.txt and buy.txt.

In there:

* The price.txt file contains information about products, including: The first line contains a positive integer N (N <= 20) which is the number of products currently in the store. The next N lines each contain 2 positive integers that are the IDs and prices of the respective products

     Example:

          3

          1 10

          2 15

          3 12

Meaning: currently there are 3 products for sale in the store, product 1 costs 10 products 2 costs 15 and product 3 costs 12.

* The buy.txt file contains information about the customer's purchase of goods. The first line contains a positive integer M representing the number of purchases made in a day. The next M lines each contain the following information: there will be multiple values ​​in a row, separated by a space. The first value will be the customer name (in string form), the remaining values ​​will come in pairs, respectively: <product ID>\_<corresponding quantity>

     Example:

          2

          A 1 2 2 3

          B 1 3 3 2

It means that there are 2 customers A and B, A buys 2 products of type 1 and 3 products of type 2; B buys 3 products of type 1 and 2 products of class 3.

Note: Product ID and customer name are unique (no repetition).

Calculate how much each customer needs to pay, then output the results to the screen.

**Input:**

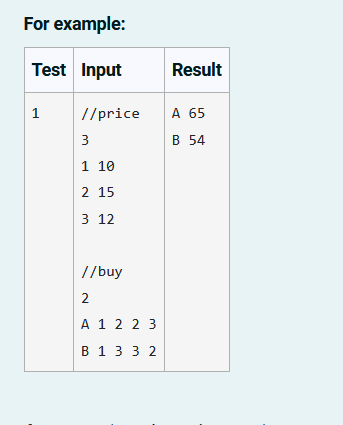
2 variables "price" (the name of the file containing product information) and "buy" (the name of the file containing purchase information).

**Output:**

Amount each customer needs to pay.

For example:

| **Test** | **Input** | **Result** |
| --- | --- | --- |
| 1 | //price  3  1 10  2 15  3 12  //buy  2  A 1 2 2 3  B 1 3 3 2 | A 65  B 54 |



void calMoney(string price, string buy) {

// TODO

}

### [Q 4.7] library management

Viết hàm **void manage(string library, string book, string author)** đọc vào 3 file library.txt và book.txt và author.txt.

Trong đó:

* File library.txt chứa thông tin của các thư viện, gồm: dòng đầu tiên chứa số nguyên dương N là số lượng thư viện được khảo sát. N dòng tiếp theo, mỗi dòng chứa 6 giá trị được phân cách nhau bằng dấu khoảng trắng. Cho mỗi dòng, giá trị đầu tiên là tên của Thư viện (tên Thư viện là duy nhất), 5 giá trị còn lại là 5 số nguyên dương, là ID của 5 quyển sách có trong thư viện đó.

     Ví dụ:  
          5

LA 1 2 3 4 5

          LB 5 3 1 2 4

          LC 4 1 5 2 3

* File book.txt chứa thông tin của các quyến sách, gồm: dòng đầu tiên chứa số nguyên dương M là số lượng đầu sách có trong tất cả các thư viện. M dòng tiếp theo mỗi dòng chứa 3 giá trị (phân cách nhau bởi một dấu khoảng trắng) có ý nghĩa như sau: giá trị đầu tiên là một số nguyên dương đại diện cho Mã số sách (ID - ID là duy nhất), giá trị thứ 2 là năm sản xuất và giá trị cuối cùng là thể loại.

     Ví dụ:  
          5

1 2000 A

          2 2001 B

          3 1993 D

          4 1997 A

          5 1995 C

* File author.txt chứa thông tin của các tác giả, gồm: dòng đầu tiên chứa số nguyên dương P là số lượng tất cả các tác giả của các sách trong các thư viện (giả sử 1 quyển sách chỉ được sáng tác bới 1 tác giả). P dòng tiếp theo mỗi dòng chứa các giá trị như sau (các giá trị được phân cách với nhau bằng 1 dấu khoảng trắng): giá trị đầu tiên là Tên tác giả, các giá trị còn lại là ID của các quyển sách mà người đó đã sáng tác.

     Ví dụ:

          3

          David 1 5

          John 3

          Henry 2 4

Xác định xem Thư viện L có chứa tác phẩm nào của Tác giả A hay không, nếu có xuất ra màn hình giá trị "True", ngược lại xuất ra "False". Với L và A được nhập vào từ bàn phím.

**Đầu vào:**

3 biến library, book và author lần lượt chứa tên file library.txt và book.txt và author.txt.

**Đầu ra:**

"True" hoặc "False" ứng với đầu vào.

Write a function **void manage(string library, string book, string author)** that reads into 3 files library.txt and book.txt and author.txt.

In there:

* The file library.txt contains information about the libraries, including: the first line contains a positive integer N which is the number of libraries surveyed. The next N lines each contain 6 values ​​separated by a space. For each row, the first value is the name of the Library (the name of the Library is unique), the other 5 values ​​are 5 positive integers, which are the IDs of the 5 books in that library.

     Example:

          3

          LA 1 2 3 4 5

          LB 5 3 1 2 4

          LC 4 1 5 2 3

* The file book.txt contains information about the books, including: the first line contains a positive integer M which is the number of titles in all libraries. The next M lines each contain 3 values ​​(separated by a space) that have the following meaning: the first value is a positive integer representing Book Number (ID - ID is unique), the 2nd value is the year of manufacture and the last value is the category.

     Example:

          5

          1 2000 A

          2 2001 B

          3 1993 D

          4 1997 A

          5 1995 C

* The author.txt file contains information about the authors, including: the first line contains a positive integer P which is the number of all the authors of the books in the library (assuming a book is authored by only one author. fake). The next P lines each contain the following values ​​(values ​​are separated by a space): the first value is the Author's Name, the remaining values ​​are the IDs of the books that the author has read. that was composed.

     Example:

          3

          David 1 5

          John 3

          Henry 2 4

Determines if Library L contains any works by Author A, if so, outputs "True" to the screen, otherwise outputs "False". With L and A entered from the keyboard.

**Input:**

The 3 variables "library", "book" and "author" contain the names of the 3 files "library.txt" and "book.txt" and "author.txt" respectively.

**Output:**

"True" or "False" for the input.

For example:

| **Test** | **Input** | **Result** |
| --- | --- | --- |
| 1 | LA Brian | False |
| 2 | LA David | True |

void manage(string library, string book, string author){

// TODO

}