Trạng thái	Đã xong
Bắt đầu vào lúc	Thứ Tư, 16 tháng 4 2025, 4:27 PM
Kết thúc lúc	Thứ Tư, 16 tháng 4 2025, 4:45 PM
Thời gian thực hiện	18 phút 5 giây

Câu hỏi 1

Đúng

Đạt điểm 1,00

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.

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	Input	Result
1	2 8 8 8 6 9 9 9 5	A 1 B 1 C 0 D 0

Answer: (penalty regime: 0 %)

Reset answer

```cpp
#include <iostream>
#include <fstream>
#include <string>

void studentGrading(string fileName) {
    // Mở file để đọc
    ifstream file(fileName);

    // Kiểm tra xem file có mở thành công không
    if (!file.is_open()) {
        cout << "Không thể mở file: " << fileName << endl;
        return;
    }

    // Đọc số lượng sinh viên
    int N;
    file >> N;
    // Biến đếm số lượng sinh viên mỗi loại
    int countA = 0, countB = 0, countC = 0, countD = 0;

    // Đọc điểm của từng sinh viên và phân loại
    for (int i = 0; i < N; i++) {
        double nmdt, ktlt, dsa, ppl;
        file >> nmdt >> ktlt >> dsa >> ppl;

        // Tính điểm trung bình
        double dtb = (nmdt + ktlt + dsa + ppl) / 4.0;
        // Kiểm tra có môn nào dưới 5 không
       bool hasFailedSubject = (nmdt < 5.0 || ktlt < 5.0 || dsa < 5.0 || ppl < 5.0);
        // Phân loại sinh viên
        if (dtb >= 8.0 && !hasFailedSubject) {
            countA++;
        } else if (dtb >= 6.5 && dtb < 8.0 && !hasFailedSubject) {
            countB++;
        } else if (dtb >= 5.0 && dtb < 6.5 && !hasFailedSubject) {
            countC++;
        } else {
            countD++;
        }
    }
    // Đóng file
    file.close();
    // Xuất kết quả
    cout << "A " << countA << endl; // Số lượng sinh viên xếp loại A
    cout << "B " << countB << endl; // Số lượng sinh viên xếp loại B
    cout << "C " << countC << endl;
    cout << "D " << countD << endl;
}
```

	Test	Input	Expected	Got	
✓	1	2 8 8 8 6 9 9 9 5	A 1 B 1 C 0 D 0	A 1 B 1 C 0 D 0	✓

Passed all tests! ✓

Câu hỏi **2**

Sai

Đạt điểm 1,00

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

Answer: (penalty regime: 0 %)

Reset answer

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

void calMoney(string price, string buy) {
    // Read product prices from price.txt
    ifstream priceFile(price);
    if (!priceFile.is_open()) {
        cout << "Cannot open file " << price << endl;
        return;
    }

    // Read number of products
    int n;
    priceFile >> n;

    // Create map to store product prices
    unordered_map<int, int> productPrices;
    for (int i = 0; i < n; i++) {
        int id, productPrice;
        priceFile >> id >> productPrice;
        productPrices[id] = productPrice;
    }
    priceFile.close();

    // Read purchase information from buy.txt
    ifstream buyFile(buy);
    if (!buyFile.is_open()) {
        cout << "Cannot open file " << buy << endl;
        return;
    }

    // Read number of customers
    int m;
    buyFile >> m;
```

```
39      buyFile.ignore(); // Skip newline after m
40
41      // Process each customer's purchase
42      for (int i = 0; i < m; i++) {
43          string line;
44          getline(buyFile, line);
45          stringstream ss(line);
46
47          string customerName;
48          ss >> customerName;
49
50          int totalCost = 0;
51          int productId, quantity;
52
```

	Test	Input	Expected	Got	

✗	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
				0
				A 65
					✗

Some hidden test cases failed, too.

Show differences

Câu hỏi **3**

Đúng một phần

Đạt điểm 1,00

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 quyển 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 bởi 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 the number of libraries surveyed. The next N lines each contain 6 values separated by a space. For each row, the fi 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

Answer: (penalty regime: 0 %)

Reset answer

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

void manage(string library, string book, string author) {
    // Leer archivo de bibliotecas
    ifstream libraryFile(library);
    if (!libraryFile.is_open()) {
        cout << "No se puede abrir el archivo: " << library << endl;
        return;
    }

    // Leer número de bibliotecas
    int N;
    libraryFile >> N;
```

```
20
21      // Crear mapa: nombre de biblioteca -> conjunto de IDs de libros
22      map<string, set<int>> libraryBooks;
23      for (int i = 0; i < N; i++) {
24          string libraryName;
25          libraryFile >> libraryName;
26
27          // Leer los 5 IDs de libros para esta biblioteca
28          set<int> books;
29          for (int j = 0; j < 5; j++) {
30              int bookId;
31              libraryFile >> bookId;
32              books.insert(bookId);
33          }
34
35          libraryBooks[libraryName] = books;
36      }
37      libraryFile.close();
38
39      // Leer archivo de autores
40      ifstream authorFile(author);
41      if (!authorFile.is_open()) {
42          cout << "No se puede abrir el archivo: " << author << endl;
43          return;
44      }
45
46      // Leer número de autores
47      int P;
48      authorFile >> P;
49
50      // Crear mapa: nombre de autor -> conjunto de IDs de libros
51      map<string, set<int>> authorBooks;
52      for (int i = 0; i < P; i++) {
53          string authorName;
54          authorFile >> authorName;
55
56          // Leer todos los IDs de libros para este autor
57          set<int> books;
58          int bookId;
59          // Leer hasta el final de la línea
60          while (authorFile >> bookId) {
61              books.insert(bookId);
62
```

	Test	Input	Expected	Got	
✓	1	LA Brian	False	False	✓
✓	2	LA David	True	True	✓

Your code failed one or more hidden tests.