File Organization

Task Description

In this lab session, we are going to implement simple operations related to file organization.

There are 2 important classes in this problem, namely `File` and `Folder`. For simplicity, we assume that a folder may contain any number of files but not a folder. Users can:

1. Create files and folders
2. Delete a file from a folder
3. Move a file to other folder
4. Count the size of a folder
5. Find the folder with largest size

In the `File` class, you may consider ***name*** (the name of the file), ***size*** (the file size) and ***foldername*** (the folder in which the file resides) as the attributes.

In the `Folder` class, you may consider ***name*** (the name of the folder) and ***listOfFiles*** (the list of files that are in the folder) as the attributes.

Input

The first line of the input contains an integer Q ($Q \leq 100$), which is the number of operations.

Each of the next Q lines is in one of the following forms:

1. **Createfile N S F**. A file with name **N** and size **S** is created inside folder with name **F**. You may assume that the folder with name **F** has been created before this operation is called.
2. **Createfolder N**. A folder with name **N** is created.
3. **Deletefile N**. Delete a file with name **N**.
4. **Movefile N F**. Move a file with name **N** to a folder with name **F**.
5. **Count F**. Return the total size of files in folder **F**. Output the size of folder **F** for this operation.
6. **Findlargest**. Return the folder with the largest size. Output the name of the folder with largest size for this operation. It is guaranteed that there is only one largest folder.

Assumption:

1. You may assume that each file has a unique name.
2. You may assume that each folder has a unique name.
3. You may assume that each operation is valid. For example, when you move a file to a folder, you may assume that that file and folder exist, i.e. they have been created from previous operation.

Output

For each **Count** and **Findlargest** operations, output the result.

Sample Input	Sample Output
12	30
Createfolder lab	15
Createfile lab1 10 lab	25
Createfile lab2 20 lab	tutorial
Createfolder tutorial	lab
Createfile tutorial1 15 tutorial	
Count lab	
Count tutorial	
Movefile lab1 tutorial	
Count tutorial	
Findlargest	
Move tutorial1 lab	
Findlargest	

Skeleton program

1. File.h

```
#ifndef file_h
#define file_h

#include <iostream>
#include <cstdlib>
#include <string>
using namespace std;

class File {
      private:
      string name;
      int size;
      string foldername;

      public:
          File();
          File(string name, int size, string foldername);

          string getName();
          int getSize();
          string getFolderName();

          void setFolderName(string foldername);
};

#endif
```

2. File.cpp

```
#include "File.h"

File::File() {
}

File::File(string name, int size, string foldername) {
}

string File::getName() {
	return "";
}

int File::getSize() {
	return 0;
}

string File::getFolderName() {
	return "";
}

void File::setFolderName(string foldername) {
}
```

3. Folder.h

```
#ifndef folder_h
#define folder_h

#include <cstdlib>
#include <cstring>
#include <vector>
#include "File.h"
using namespace std;

class Folder {
	private:
		string name;
	vector<File*> files;

	public:
	    Folder();
		Folder(string name);

	string getName();

	void addFile(File* file);
	void deleteFile(File* file);
	int countSize();
};

#endif
```

4. Folder.cpp

```
#include "Folder.h"

Folder::Folder() {
}

Folder::Folder(string name) {
}

string Folder::getName() {
    return "";
}

/**
*     Add a file to the folder
*     Pre-condition  :
*     Post-condition :
*/
void Folder::addFile(File* file) {
}

/**
*     Delete a file from a folder
*     Pre-condition  :
*     Post-condition :
*/
void Folder::deleteFile(File* file) {
}

/**
*     Count the total size of a folder
*     Pre-condition  :
*     Post-condition :
*/
int Folder::countSize() {
      return 0;
}
```

5. main

```
/**
*     Name              :
*     Matric-number     :
*     Plab account      :
*/
#include <iostream>
#include <cstdio>
#include "Folder.h"
using namespace std;

/**
*     Find largest folder from a list of folders
*     Pre-condition  :
*     Post-condition :
```

```
*/
string findLargest(vector<Folder*> folders) {
      return "";
}

int main() {
    // read input

    // process the input

    // output
    return 0;
}
```

Note: You will not be penalized if you did not program in OOP. But make sure that your program is modular. Please be reminded that the marking scheme is **Input:10%, Output:10%, Programming Style:30% and Correctness: 50%**