Data Structures and Algorithms I
Spring 2024
Programming Assignment #1

You are going to write a program that keeps track of mysterious creatures known as foobars. Every foobar has a name. For the purposes of this assignment, each name will be a string consisting of uppercase and lowercase letters and digits. It is possible that multiple foobars will have the same name.

Foobars like to line up; when they do, it gives them strength. Foobars who are not standing in a line have zero strength. Ordinary foobars who are standing in a line have a strength equal to their position in the line. For example, if they are at the front of the line, their strength is 1, and if they are at the back of the line, their strength is N, where N is the number of foobars in the line. Some foobars have special status. For example, some foobars are foos. A foo's strength is equal to their position in the line multiplied by 3. Some other foobars are bars. A bar's strength is equal to their position in the line plus 15. (Some foobars are neither foos nor bars; these are what I referred to as "ordinary foobars" earlier. No foobar is both a foo and a bar.)

Your program will read a sequence of foobars from a text file. Each line represents one foobar and contains two words. The first word will be "foobar", "foo", or "bar", representing the type of foobar. The second word will be the name of the foobar. The information in the file represents a sequence of foobars standing in a line. The first line of the file represents the foobar at the back of the line; the last line of the file represents the foobar at the front of the line. There will be no leading or trailing whitespace, a single space between words, and a Unix-style newline character at the end of each line of the file.

I will provide you with a text file called "sampleInput.txt", with the following content (some of the names may seem odd):

```
foobar R2D2
foo Carl
foo Algorithms
bar IAmABar
bar xyz
foo 123q456
foo CooperUnion
bar DataStructures
foobar xyz
foo CooperUnion
```

Your program should ask the user the for the name of the input file (e.g., "sampleInput.txt") and the name of an output file (e.g., "sampleOutput.txt"). Do not hardcode filenames in your code. The output file should store information about the same foobars as the input file in the same order. Each line of the output file should list the name of the foobar (which might be a foo or a bar), followed by a single space, followed by their strength, followed by a single Unix-style newline character. A sample run of the program might look simply like this:

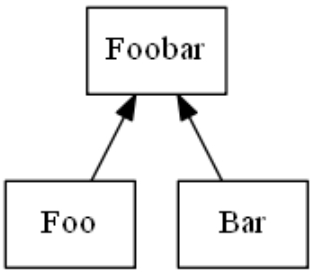
Data Structures and Algorithms I
Spring 2024
Programming Assignment #1

```
Enter the name of the input file: sampleInput.txt
Enter the name of the output file: sampleOutput.txt
```

If you process the given file "sampleInput.txt", and store the results in "sampleOutput.txt", as indicated in the sample run, then the contents of "sampleOutput.txt" should look exactly like this:

```
R2D2 10
Carl 27
Algorithms 24
IAmABar 22
xyz 21
123q456 15
CooperUnion 12
DataStructures 18
xyz 2
CooperUnion 3
```

I am requiring that you store the information about each foobar in an object. Since every foo is a foobar, and every bar is a foobar, it makes sense to use a class hierarchy. A simple class diagram for such a hierarchy might look like this:



Note that Foobar is the base class, a.k.a. superclass (not abstract, since some foobars are neither foos nor bars). Foo and Bar are derived classes, a.k.a. subclasses. The name of each foobar and its current position in a line should be stored in the base class as private data members. The base class should also have public member functions to set the position in a line, to get the name, and to get the strength (which is based on the position in the line). The member function to get the strength should be virtual, and it should be overridden in the derived classes. There should also be a protected member function to get the position (the derived classes will need this to determine the strength). The name should only be set once, in the base class's constructor. The constructor should initially set the position to 0, meaning that the foobar is not currently in a line.

Your program should loop through the input file only once, in a single pass, adding information about each foobar into a vector of pointers to Foobar objects. You can use the vector class's

Data Structures and Algorithms I
Spring 2024
Programming Assignment #1

"push_back" member function to add each pointer to the vector. (Although push_back is worst-case linear, because sometimes the vector needs to grow larger, it is implemented in such a way that it has amortized constant time.) After processing the input, your program should then loop through the vector backwards to update the position of each foobar (the final foobar in the vector is in line position 1, indicating the front of the line). Then, the program should loop through the vector forwards to display the information about each foobar to the output file.

Note that I am not claiming this is the easiest way to implement this program; it almost certainly is not. The goal here is to force you to gain familiarity with various, important, object-oriented concepts that will be useful for later programs. Here are some additional suggestions:

- I'm allowing you to place all your code in a single file, including your class definitions, the member function implementations, and the regular functions that implement your program. However, don't put all your code in a single function. I suggest using separate functions to read the input file, to update the strength of foobars, and to write the output file. In particular, it is considered good convention to keep the "main" function of a program short.
- Don't use unnecessary global variables. Declare your main vector locally in "main" and pass it to other functions that use it by reference.
- In the function that processes the input file, read each line of the input file one line at a time, using the provided "getline" function (part of the C++ standard library). *I guarantee that the input file will be properly formatted* (your code does not have to check).
- Create an object of type "stringstream" (a provided class) for each line of input, and then read the words (indicating the type and name of the creature) from it just as if you were reading from standard input. Then, dynamically allocate a foobar, foo, or bar object, and add the pointer to it to your vector. When you allocate the object, pass the constructor the name of the foobar (the initial position should have a default value of 0).
- After processing the input file, call another function to loop through the objects in the vector backwards, updating the line positions appropriately. Note that the strengths should not be stored in the objects; they are based on the position, which is stored.
- In another function, loop through the vector in order, invoke appropriate member functions to get the names and strengths of the foobars, and write this information to the output file.

Your grade will depend not only on correctness, but also style, elegance, comments, formatting, adherence to proper C++ conventions, and anything else I think constitutes good programming. You should include one comment at the top of your code with your name and a brief description of what the program does. You should also include comments above functions, class definitions, non-trivial member functions (either by their declarations or their implementations), and anywhere else that you think the code is doing something that is not obvious. Comments should be brief and should not state something that is obvious. *This is an individual programming assignment.* It is fine to ask each other questions, but you should not share code.

Data Structures and Algorithms I
Spring 2024
Programming Assignment #1

Submit your program by e-mail to *carl.sable@cooper.edu*. Send your source code only (*no executables or object files*) as an attachment. I will compile your program and test it on my test cases. I will not release my test cases to students; they will be much larger than the provided test case. (My largest input file contains millions of lines; I guarantee it will be small enough such that strengths of foobars do not overflow 32-bit integers.) *The assignment is due before midnight on the night of Tuesday, March 12*. You may send me early pre-submissions, and I will let you know if they are working for my test cases, but I won't look over your code until I am grading your final submissions. (It will typically take up to 48 hours to reply to pre-submissions, so you need to send them at least that long before the deadline to guarantee a reply.) I advise you to get started early, especially if this is your first time writing an object-oriented program.