CSCI 235 – Software Design & Analysis II
Assignment 1

Introduction

Before starting this assignment, read the following programming rules:
http://www.compsci.hunter.cuny.edu/~sweiss/course_materials/csci235/programming_rules.pdf

This assignment tests your ability to design and implement functional modules in a working C++ program with classes and file I/O. The program must compile and run on the computers in 1000G HN (G-lab). You may work on this program by yourself or in a group. (If you choose to work in a group, each member receives the same grade. The maximum group size is 3 and must be composed of other students taking CSCI 235 this semester.)

Assignment

The attached file lists muscles in the human body and some of their properties, including “Origin” (where the muscles attach) and Action (what the muscle does). The file is in CSV format. You should be able to find web pages about CSV files and should be open the file with a spreadsheet program, like Excel or OpenOffice Calc. The first row or line is a description for the rest of the rows or lines in the file.

Write a program which reads the attached file and allows the user to receive an answer to certain queries. Your program must support Action Queries, Strengthening Queries and Diagnostic Queries.

When executing an Action query, the user will type one of the verbs defined, such as:

- extend
- flex
- raise
- close

… or any of the other actions as defined by the file's “Action” column. In addition, the user will add a body part associated with the Action. The program must respond with the muscle associated with the action. For example, if the user performs the following query:

```
compress cheek
```

… the program must respond:

```
The buccinator muscle is involved
```

Notice that the file's Action column is written in the 3rd person singular (“compresses cheek), but your program must allow queries in the imperative (“compress cheek”). Once the program accounts for this difference, it may search only the Action column for this query. If no muscle is associated with the action, the program must indicate that the action is not possible.

With Strengthening Queries, the user will type the word “strengthen” along with a body part, and the program must respond with all the muscles which have the body part listed in the “Location” column. For example, if the user performs the following query:

```
strengthen neck
```

… the program must respond:

```
Work on the following muscles: platysma, digastricus and
sternocleidomastoid
```

The list of muscles may be ordered in any manner you find convenient.

A Diagnostic Query gives the user information on orders obtained from a physical therapist. The query returns the body part (listed in the "Location" column) associated with a muscle (listed in the "Muscle" column) and its depth (listed in the "deep / superficial" column) when the user provides the name of the muscle. For example if the user issues the following query:

```
teres major
```

… the program must respond:

```
possible deep shoulder pain
```

Submission

Submit your UML design and source code on Blackboard. In the submission comments, list all people who worked on the code. If you work in a group, only one team member needs to submit for the group.

Submit all your code and documentation as one "tar.gz" file. A tar file concatenates a bunch of different files (without compressing them). A gz (gzip) file compresses a single file. You can create a tar file (named "a1.tar") in the same directory from three files (main.cpp This.hpp This.cpp) with the following command:

```
tar -cvf ./a1.tar main.cpp This.hpp This.cpp
```

You can gzip the tar file above with the following command:

```
gzip a1.tar
```

This creates a file in the same directory called "a1.tar.gz", which is what you should submit on Blackboard.

Grading

Your grade will be based on the following:
50% = Correctness
 The program must compile and run on one of the G-lab machines. In addition, it must perform the functions outlined in the "Assignment" section.
25% = Design
 The program must show a reasonable object-oriented decomposition of the assignment into classes. The UML must match the implemented program.
15% = Performance
 The implementation must be as efficient as possible in terms of the amount of memory used and in terms of the amount of computational cycles used.
10% = Documentation and style
 The implementation must have good comments; variables must have reasonable names, and the submission must have instructions with respect to compilation and execution. The package must be placed on Blackboard as outlined in the "Submission" section.
There is a 10% late penalty per day after the first day.