Problem Set 4

Due before midnight on Monday, October 17, 2016.

This assignment is designed to give you experience in refactoring existing code. It will also be good preparation for the midterm since it will encourage you to look closely at some of the demo code from class.

1 Assignment Goals

Example 08-BarGraph has been extensively discussed in class. Several design issues, bugs, and places for improvement were mentiond. The process of taking an existing piece of code and modifying it to improve its design and style while preserving functionality is called *refactoring*. The purpose of this assignment is to give you a deeper understanding of the many design decisions that went into the relatively simple bar graph program, and to experience what is involved in refactoring.

In particular, any changes are likely to result in new compiler errors as variables are renamed, types are changed, and code is moved around. Since you are starting with code that compiles, a compiler error that results from a change you made shows you where to look for the problem. You should work a little bit at a time and recompile after each related group of changes to make sure you haven't broken things.

2 Problem

You should make the following changes in `08-BarGraph`:

1. (4 points) The code that opens the input file and reads it used to be split between `run()` and the `Graph` constructor. The body of `run()` should be moved to a class function in a new class `Controller`. The file reading code in the `Graph` constructor should also be moved to one or more class functions in `Controller`. The modified `run()` in `main.cpp` should instantiate `Controller` and pass control to a function in it. The function `Graph::insert()` will need to be made public.

2. (1 point) In `row.hpp`, class `Cell` should be eliminated. The row should be represented variable of type `vector<Item>` instead of my a linked list of `Cell`.

3. (3 points) In `graph.hpp`, the element type of array `bar` should be changed from `Row*` to `Row`. This will have ramifications elsewhere in your program. In particular, `Row::insert()` should call `vector<Item>::push_back(Item(name, score))` in order to create and insert a new `Item` into a row rather than using `new` to create an `Item*`.

4. (1 point) You should initialize the `bar` array by declaring it with the initializer list
   `= {0,1,2,3,4,5,6,7,8,9,10}`.

5. (2 points) The potential infinite loop in the `Graph` constructor should be fixed before it is moved to the `Controller` class.

6. (1 points) Every use of C-style strings (`char*` or `char[]`) should be changed to use a C++ `string` instead. (The one exception is `argv` in `main()`, which should remain a `char*` array.)

7. (2 points) In `row.cpp`, the cleverness in constructing the label string should be replaced by a straightforward use an `ostringstream`. First figure out what you would have to do to print the label for row `rowNum` if you were just printing to `cout`. Then "print" the same way to an `ostringstream` and use the ostringstream function `str()` to obtain the underlying string.

8. (1 point) You should eliminate `rowNest.hpp` and the conditional code that depends on the `NEST` preprocessor variable, and make corresponding changes to `Makefile`.

9. (1 point) With the above changes, there should be no further occurrences of `new` or `delete` in your program, so remove all destructors from your code as well as any constructors that don't do anything useful.

Grading Rubric

Your assignment will be graded according to the scale given in Figure 1 (see below).

#	Pts.	Item
1.	1	All relevant standards from PS1 are followed regarding submission, identification of authorship on all files, and so forth.
2.	1	A well-formed `Makefile` or `makefile` is submitted that specifies compiler options `-O1 -g -Wall -std=c++14`.
3.	1	Running `make` successfully compiles and links the project and results in an executable file `bar`.
4.	1	Your program gives the same output as 08-BarGraph except for the debugging comments written to `cerr`, which should be eliminated.
5.	16	Each of the changes required section 2 is worth the indicated number of points.
	20	Total points.

Figure 1: Grading rubric.