Assignment 1

`Answers.txt` Due: 10:00 PM, Tuesday, September 9, 2003.
Runnable Code Due: 10:00 PM, Wednesday, September 10, 2003.
Final Code Due: 10:00 PM, Thursday, September 11, 2003.

Preliminaries

As with all assignments in this course, part of your grade will be based on your coding style and on the way your written work is presented. The *Homework Policies* handout discusses these issues in depth. The *Electronic Submission* handout explains how to submit your work and how to use the submission system to facilitate working in pairs. If you do not have copies of these handouts, you can obtain a copy by visiting the course website at http://www.cs.hmc.edu/cs70/. Remember, you should work on this entire assignment as a pair, including the written part. You can read the assignment and think about it before you meet, however.

Before You Begin

Download the sample source code included with this assignment from the Homework area of the course website. The file archive to download is `assign1.tgz`. On *turing* you can download and unpack the file archive by executing

```
cs70checkout cs70ass1
```

This command should check out the following files and directories:

```
cs70ass1/SCHEDULE
cs70ass1/Answers.txt
cs70ass1/palindromes/Makefile
cs70ass1/palindromes/Palindrome.java
cs70ass1/palindromes/palindrome.cpp
cs70ass1/shuffler/Makefile
cs70ass1/shuffler/LineShuffler.java
cs70ass1/shuffler/shuffle.cpp
cs70ass1/shuffler/lineshuffler.cpp
cs70ass1/shuffler/lineshuffler.hpp
cs70ass1/shuffler/random.cpp
cs70ass1/shuffler/random.hpp
```

The `palindromes` directory contains C++ and Java implementations of a program that determines whether a word entered by the user is a palindrome (the same forwards as backwards). Both programs can be built by entering the `palindromes` directory and typing `make` (the same command will rebuild the executables if you edit the files). The C++ version can be run by typing `./palindrome`; the Java version can

be run by typing `java Palindrome`. Both implementations should operate identically.

The `shuffle` directory contains C++ and JAVA implementations of a program that reads several lines from the user and the outputs them in random order. Both programs can be built by entering the `shuffle` directory and typing `make`. The C++ version can be run by typing `./shuffle`; the JAVA version can be run by typing `java LineShuffler`. Both implementations should operate identically.

Written Questions

Answers to the questions in this section should be one sentence (or, in some cases, one line of code) only. You can make modifications to the code for the assignment to answer these questions, but should always return the code to its previous (working) state before moving on to the next question.

You should provide your answers by adding them to the file Answers.txt, provided in the assignment file archive. You should submit your answers using the `cs70submit` command. You may revise and resubmit your answers as many times as you wish until 10:00 PM on Tuesday. Answers submitted after this deadline will be ignored.

W1. The C++ version of the palindrome program does not define a class, whereas the JAVA version does (defining a class called *Palindrome*).

   (a) When the JAVA version executes, how many objects of class *Palindrome* are created? (If you are not sure, you could provide a constructor for the class that outputs a message and count how many messages appear.)

   (b) The C++ version could have been written to place most of the code inside a class like the JAVA version, but wasn't. Why?

W2. When building the palindrome program (by typing `make`),

   (a) What command line is executed by `make` to compile the C++ code?

   (b) What do each of the command-line options mean? (You may want to type `man g++` to look up information, but if you do, remember that you don't need to read the entire manual page.)

W3. Both the C++ and JAVA versions of the palindrome program use a string class provided by the language (*string* in C++; *String* in JAVA). Based on the code for the palindrome program alone,

   (a) What string method behaves exactly the same in C++ and JAVA?

   (b) What string operation exists in both C++ and JAVA but is accessed differently?

W4. What is the closest equivalent in C++ to JAVA's `import` mechanism?

W5. What is the the C++ equivalent to the following JAVA code?

```
System.out.println("Hello World!");
```

W6. Rearrange the C++ file palindrome.cpp to place the function definition of main above the definition of the isPalindrome function.

(a) What compiler error is given by the compiler (g++) when you run make?

(b) Would the JAVA compiler give an error if a similar rearrangement were made in the JAVA source?

(c) Why doesn't C++ handle the code rearrangement the same way as JAVA?

(d) Adding a *function declaration* for isPalindrome can eliminate the error. What is the correct declaration?

W7. In the line-shuffling example, the JAVA code (in `LineShuffler.java`) contains the line

```
LineShuffler randomLines = new LineShuffler();
```

whereas the C++ implementation (in `shuffle.cpp`) achieves similar results using

```
LineShuffler randomLines;
```

(a) If you change the JAVA code for this declaration so that it looks like the C++ code and recompile the program, what error message does the compiler give?

(b) What error message does the compiler give if you change the C++ code to

```
LineShuffler randomLines = new LineShuffler;
```

(c) What does this error message mean?

W8. In JAVA, a class is defined by a single file (ending with `.java`), but in C++ a class is usually defined by two files, a *header file* (ending in `.hpp`) and an *implementation file* (ending in `.cpp`).

(a) What does the line #include "`lineshuffler.hpp`" actually do?

(b) Why doesn't `shuffle.cpp` contain the line #include "`random.hpp`" ?

(c) Why do the names of all the member functions (a.k.a. methods) defined in the implementation file begin with LineShuffler:: ?

W9. In the header file `lineshuffler.hpp`, the class declaration ends with }; rather than just with }.

(a) Modify the class declaration so that it ends with } `myShuffler`; and recompile the code. The compiler should pass the code as being correct, even though we would consider it bad coding style in CS 70 for a couple of reasons. (The linker, on the other hand, will generate an error when it tries to make an executable from the object files produced by the compiler.)

i. What does this code mean?

N.B. You almost certainly won't find an answer to this question in your textbook. Investigate for yourself. Try `x, y, z` instead of `myShuffler`. Consider what it *could* mean, and then test your guess.

ii. Aside from the odd-looking syntax, why is this code bad style?

(b) Modify the class declaration so that it ends with } (i.e., no closing semicolon) and attempt to recompile two of the source files individually by typing `make shuffle.o` and `make lineshuffler.o`.

i. What errors does the compiler report for each of these files?

ii. One of the files gives a sensible error message, whereas the other file gives a rather obscure message. This latter error shows the compiler is fairly confused. What does it think you are trying to do and why?

W10. The constructor for *LineShuffler* class contains the line

```
: count_(0)
```

(a) What does this part of the code mean?

(b) What would happen to the data member `count_` if it were omitted?

W11. The `count_` data member of the *LineShuffler* class is redundant. Explain why by indicating how the number of lines stored in the *LineShuffler* can be quickly determined.

Coding Question

The palindrome program has several limitations. The foremost is that the `isPalindrome` function was designed only to check whether a single word is a palindrome, rather than a whole sentence. Thus while it happily confirms that words like "detartrated", "repaper", "rotator", "kayak" and "civic" are palindromes, it spurns palindromic sentences such as "Lived on decaf, faced no devil..." and "Was it a car or a cat I saw?" because

1. The palindrome test is case sensitive
2. The palindrome test does not ignore non-letters (spaces, punctuation, etc.)

Fix these deficiencies and submit your new code containing a revised version of isPalindrome. To help you, you may use two functions provided by C++ (accessible after adding the line #include <cctype> to the #includes at the top of your file). The functions are

```
char tolower(char c);          // Returns c, converted to lower case.
bool isalpha(char c);          // Returns true, if c is a letter.
```

You should submit your code using the electronic submission system described in the *Electronic Submission* handout. You are not required to include a README file with this assignment.