

CS 215 - Fundamentals of Programming II

Spring 2008 - Project 5

20 points

Out: March 17, 2008

Due: March 31, 2008

Note that although this project is not due until after Exam 2, the material covered in this project will be on Exam 2.

This project consists of writing two short applications, one that uses STL vectors and STL lists and one that uses STL queues.

Part I: Creating a concordance

(10 points) Since a list must be searched sequentially, when a list grows to be large, finding a value in the list is not efficient. One way to improve efficiency is to maintain several smaller lists. For example, a text concordance is a list of all distinct words appearing in a particular text. Write a program in the file `concordance.cpp` that reads words in lowercase (only) text and produces a text concordance. (We will ignore uppercase, numerals, and punctuation for this program.) The program must meet the following criteria:

- The concordance must be implemented as a vector with 26 elements that are lists of strings. All words beginning with **a** should be stored in the first list of the vector (i.e., the list at index 0), those beginning with **b** should be stored in the second list of the vector, and so on.

A side note regarding syntax: normally the compiler does not care about whitespace in instantiating a template class, so to declare a vector of lists of strings, we might type:

```
vector <list <string>> vectorOfStringLists;
```

But when we do this, the compiler will interpret the “>>” as an attempt to use the input operator. So when you instantiate a template class with another template class, you must put a space between the two >'s and type:

```
vector <list <string> > vectorOfStringLists;
```

- The concordance is to be written in alphabetical order to an output file, one word on a line, **directly from the vector of string lists data structure**. Prior to writing out the concordance, each list in the vector must be sorted and without duplicates. Note there are several ways to do this and you are free to do this in anyway you see fit as long as it does not involved other data structures. (E.g., you cannot pull all of the elements of a list into a vector and call one of the sorting routines from Project 3, then put them back into the list.)

This program must take two command-line arguments, the input file name and the output file name. You will write your own test files. **You must submit 1 non-trivial test file.** By non-trivial, we mean a test file of at least several lines, with words beginning with a variety of letters, and with duplicates. Please note that there should be no punctuation, no digits, and no uppercase. Name this file `sampletext.txt`.

Part II: Compete card game

(10 points) Compete is a 2-player card game similar to the game of War. Each player has a pile of cards and both piles start with the same number of cards. Play consists of a sequence of "rounds." Play continues until at the end of a round at least one player is out of cards. A discard pile is created and used during a round and is empty at the beginning of each round. A round is played by executing the following steps until the round ends:

1. Each player turns over the top card of his/her pile.
2. If the numeric value (i.e., suit is not considered in this game) of one of the cards is greater than the other, then the player with the higher card adds all the cards from the discard pile to the bottom of his/her pile without changing the order, then adds both cards just played to the bottom of his/her pile, the higher card first, and the round ends.
3. If the numeric values of both cards are the same, then both cards are placed on a discard pile that is a queue. (Note it doesn't matter what order the cards are put on the discard pile, since they have the same numeric value.)
4. Afterward, if exactly one of the player's pile is empty, then the other player adds all of the cards from the discard pile to the bottom of his/her pile without changing the order, and the round (and game) ends. The player with all the cards is the winner.
5. If both players' piles are empty at the same time, then both players' piles remain empty and the round (and game) ends. The game ended in a tie.
6. Otherwise, play for the round returns to Step 1.

Assume playing cards have order values from lowest to highest: Two, Three, Four, Five, Six, Seven, Eight, Nine, Ten, Jack, Queen, King, and Ace.

Write a program in file `compete.cpp` that plays Compete given the starting piles of two players and determines who wins. **The piles must be implemented using STL queues.** The program must accept an input file name as a command line argument. The input file will have the following format:

```
# of games in file
# of cards in starting piles of game 1
card character values in player 1 starting pile for game 1
card character values in player 2 starting pile for game 1
repeat # of cards in each pile and player starting piles for rest of games
```

A card **character** value is a single character representing the value of a card. They are the characters `2' through `9' for card values Two through Nine, `T' for Ten, `J' for Jack, `Q' for Queen, `K' for King, and `A' for Ace. You may assume there are no more than 26 cards in a starting pile (52 cards total), and that each game will end.

The program will output the following for each game in the file: the game number (starting with 1), the number of rounds played, and the result (Player 1 wins, Player 2 wins, or a tie). The format must be exactly shown in the example below.

Example

The following is sample input for this problem:

```
3
2
3 5
J 5
3
3 5 7
3 5 7
5
3 5 T K 4
J 5 2 6 4
```

The following is the correct output for the input above:

```
Game 1: After 2 round(s), Player 2 wins
Game 2: After 1 round(s), the game ends in a tie
Game 3: After 11 round(s), Player 1 wins
```

You are not required to submit a makefile for either program.

REMINDER: Your project must compile for it to be graded. Submissions that do not compile will be returned for resubmission and assessed a late penalty. Submissions that do not substantially work also will be returned for resubmission and assessed a late penalty.

Follow the guidelines in the [C++ Programming Style Guideline](#) handout. As stated in the syllabus, part of the grade on a programming project depends on how well you adhere to the guidelines. The grader will look at your code listing and grade it according to the guidelines.

What to submit

Electronically submit a tarfile containing `concordance.cpp`, `sampletext.txt`, `compete.cpp` as explained in the handout [Submission Instructions for CS 215](#). Also hand in hard-copy printouts of `concordance.cpp` and `compete.cpp` (only) as explained in the handout. **Do not** submit object files or executable files.