Program 4: Wheel of Lies
CS241 Fall 2014
Due Friday December 5

Not accepted late.

Announcements/Changes

Overview

You may work on this assignment individually, or with one other person in CS 241. The goals of this assignment are to:

1. Become familiar with using maps and sets.
2. Discover how maps are implemented.
3. Create a binary search tree.

Program Specifications

The game Wheel of Fortune is based on a very old game called Hangman. In Hangman, one player (the executioner) selects a word and writes out a dash for each letter in the word. The other players (the condemned) then attempt to guess the letters in the word. If the condemned guesses a letter in the word, the executioner fills in each place in the word where this letter occurs. If the letter does not appear in the word, the executioner draws one element of a hanged man. If the condemned guesses all of the letters in the word before the picture is complete, they win. If not, the execution wins.

In this assignment, you will be making a computer program that plays Hangman (as the executioner). The interesting wrinkle is that your computer program will cheat. How? Well, when a player guesses a letter the program will break its list of words into a set of word families based on where that letter occurs in the word. The program will then select the largest word family and tell the user that the guessed letter appears where it does in the word family. In this way, the program does not commit to a specific word until it absolutely has to. Here is an example, say our word list includes on the following words:

ALLY BETA COOL DEAL ELSE FLEW GOOD HOPE IBEX

Suppose the condemned guesses the letter 'E.' Since the program hasn't chosen the word yet, there are several options for where an 'E' might appear.

ALLY B**E**TA COOL D**E**AL **E**LS**E** FL**E**W GOOD HOP**E** IB**E**X

Each word in this list belongs to a single word family:

- _ _ _ _: ALLY, COOL, and GOOD

- _ E _ _: BETA and DEAL
- _ _ E _: FLEW and IBEX
- E _ _ E: ELSE
- _ _ _ E: HOPE

Because the _ _ _ _ family contains the most words, the program would choose it and tell the condemned that 'E' does not appear in the word. The new word list would be:

ALLY COOL GOOD

If the condemned guessed 'O' next, your word list would break into two families:

- _ _ _ _, ALLY
- _ O O _, COOL and GOOD

The second family is the largest, so your program would tell the condemned that 'O' appears in the word as the 2nd and 3rd letters. The word list is now reduced to:

COOL GOOD

Say the condemned guesses 'Z'. Well, this results in only a single word family:

- _ _ _ _, COOL and GOOD

Here the choice is simple.

Eventually, it may occur that only one word remains in the list. At this point the cheating program must play according to the traditional runs of Hangman, and so it is possible for the condemned to win. If the condemned runs of guesses before filling in the word, the program should select any word from its current word list and claim that this was the secret word all along.

The high level algorithm for this program goes as follows:

1. Read in the file *lexicon.txt*. This file contains a set of over 120,000 possible words.
2. Prompt the user for a word length for the game. The program should reprompt if the user enters an invalid number like -47 or a length for which no words exist in the lexicon.
3. Prompt the user for a number of guesses. Reprompt if the number is not greater than 0.
4. Prompt the user to see if they want a running total of the number of possible words to be displayed during play.
5. Play a game of hangman using the Wheel of Lies algorithm.
    a. Construct a list of words from the lexicon that match the desired length.
    b. Print out how many guesses the player has remaining, any letters the user has guessed, and the current blanked-out version of the word. Also print the number of possible words, if desired.
    c. Prompt the user to guess a letter. Reprompt for invalid input or letters that have already been guessed.

   d. Partition the word list into word families.
   e. Select the most common (largest) word family and remove any words from the word list that aren't in the word family. If the word family contains the guessed letter, report position of the letters to the user. Otherwise, subtract a guess from the user.
   f. If the player has run out of guesses, select a word from the word list and display it as the secret word.
   g. If the player guessed the word, print a note of congratulations.
6. Ask if the player would like another game. Go to 2, as necessary.

Initially, your program will probably use STL's map class. However, before you hand it in, all instances of STL's map must be replaced by your own binary search tree implementation of a map called *LCMap*, specifically, **`LCMap<K,V,Comparator>`**. **`Comparator`** should default to STL's **`less`** comparator. Your **`LCMap<K,V,Comparator>`** should provide the following methods:

```
/* constructor */
LCMap(Comparator c = Comparator());
/* copy constructor */
LCMap(const LCMap<K,V,Comparator>&);
/* cleans up all memory for storage and calls the destructor for the
keys and values stored */
virtual ~LCMap();
/* assignment operator */
LCMap<K,V,Comparator>& operator =(const LCMap<K,V,Comparator>&);
/* inserts the key value pair */
bool insert(const K& key, const V& value);
/* erases key value pair referenced by key.
   returns true if successful */
bool erase(const K&);
/* lookup the value associated with a key. if the key is not in the
map, insert it with default value. Should provide l-value access to
value.*/
V& operator[] (const K& k);

/* returns true if this key maps to a value */
bool in (const K&);
/* returns a list of keys in this map */
list<K> keys();
/* returns true if the map is empty */
bool empty() const;
/* number of key value pairs stored */
int size() const;
/* empties the map */
void clear();
```

I will be testing your *LCMap* by loading it into a test harness. Therefore, it is vitally important that your map match the interface detailed above. If your map does not compile in the test harness you will receive a 0.

Milestones

MS1 (due Dec 1 in lab): The Wheel of Lies portion of the program must be complete and working with STL's map class. Note that this does not mark a 2/3 completion of this program. Do not try to implement *LCMap* in only five days.

How to hand in your program

Create a directory for your project with the following naming scheme: username.project (e.g., stocker_t.p4). Create a compressed tarball of this directory and submit it using Moodle.

If you are working in pairs, be sure to put both of your names in the comment section of **each** file.

Grading

Your programs will be run and graded on correctness given a variety of input parameters. They will also be graded on the correct use of data structures. Furthermore, for each assignment there will be a list of proper programming techniques that you will be graded on. For each assignment you will be responsible for ensuring that you practice the new techniques as well as the techniques outlined in all previous assignments.

Proper Programming Techniques

Be sure to follow the programming technique guides from the previous assignments. Failure to do so will result in lost points.