Final Project: Evil Hangman

Due: Thursday 11:59:59PM 4/26/2012

Objectives

Combining knowledge we have learned to implement an interesting game. Topics include File I/O, Data Structure, Loops, String and more.

Grading Rubric

Final Project is worth 20 points, and is graded with the following standard.

- 14 points **Scope and Correctness**. To what extent does your code implement the features required by the specification? To what extent is your code free of bugs?
- 4 points **Design.** To what extent is your code written well (*i.e.*, clearly, efficiently, elegantly, and/or logically)?
- 2 points **Style.** To what extent is your code readable (*i.e.*, commented and indented properly, and variables well named)

Specification

The hangman game

In case you aren't familiar with the game *Hangman*, the rules are as follows:
1. One player chooses a secret word, then writes out a number of dashes equal to the word length.
2. The other player begins guessing letters. Whenever she guesses a letter contained in the hidden word, the first player reveals each instance of that letter in the word. Otherwise, the guess is wrong.
3. There is a preset number for how many wrong guesses the guesser can have. The game ends either when all the letters in the word have been revealed or when the guesser has run out of wrong guesses.

Cheating in the hangman game

Fundamental to the game is the fact the first player accurately represents the word she has chosen. That way, when the other players guess letters, she can reveal whether that letter is in the word. But what happens if the player doesn't do this? This gives the player who chooses the hidden word an enormous advantage. For example, suppose that you're the player trying to guess the word, and at some point you end up revealing letters until you arrive at this point with only one guess remaining:
D O – B L E
There are only two words in the English language that match this pattern: “doable” and “double.” If the player who chose the hidden word is playing fairly, then you have a fifty-fifty chance of winning this game if you guess 'A' or 'U' as the missing letter. However, if your opponent is cheating and hasn't actually committed to either word, then there is no possible way you can win this game. No matter what letter you guess, your opponent can claim that she had picked the other word, and you will lose the game.
That is, if you guess that the word is “doable,” she can pretend that she committed to “double” the whole time, and vice-versa.

Let's illustrate this technique with an example. Suppose that you are playing *Hangman* and it's your turn to choose a word, which we'll assume is of length four. Rather than committing to a secret word, you instead compile a list of every four-letter word in the English language. For simplicity, let's assume that English only has a few four-letter words, all of which are reprinted here:

ALLY BETA COOL DEAL ELSE FLEW GOOD HOPE IBEX

Now, suppose that your opponent guesses the letter 'E.' You now need to tell your opponent which letters in the word you've "picked" are E's. Of course, you haven't picked a word, and so you have multiple options about where you reveal the E's. Here's the above word list, with E's highlighted in each word:

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

If you'll notice, every word in your word list falls into one of five "word families:"
- ----, which contains the word ALLY, COOL, and GOOD.
- -E--, containing BETA and DEAL.
- --E-, containing FLEW and IBEX.
- E--E, containing ELSE.
- ---E, containing HOPE.

Since the letters you reveal have to correspond to *some* word in your word list, you can choose to reveal any one of the above five families. There are many ways to pick which family to reveal – perhaps you want to steer your opponent toward a smaller family with more obscure words, or toward a larger family in the hopes of keeping your options open. In this assignment, in the interests of simplicity, we'll adopt the latter approach and always choose the largest of the remaining word families. In this case, it means that you should pick the family ----. This reduces your word list down to ALLY COOL GOOD and since you didn't reveal any letters, you would tell your opponent that his guess was wrong.

Let's see a few more examples of this strategy. Given this three-word word list, if your opponent guesses the letter O, then you would break your word list down into two families:
- -OO-, containing COOL and GOOD.
- ----, containing ALLY.

The first of these families is larger than the second, and so you choose it, revealing two O's in the word and reducing your list down to

COOL GOOD

But what happens if your opponent guesses a letter that doesn't appear anywhere in your word list? For example, what happens if your opponent now guesses 'T'? This isn't a problem. If you try splitting these words apart into word families, you'll find that there's only one family – the family ---- in which T appears nowhere and which contains both COOL and GOOD. Since there is only one word family here, it's trivially the largest family, and by picking it you'd maintain the word list you already had.

There are two possible outcomes of this game. First, your opponent might be smart enough to pare the word list down to one word and then guess what that word is. In this case, you should congratulate him – that's an impressive feat considering the scheming you were up to! Second, and by far the most common case, your opponent will be completely stumped and will run out of guesses. When this happens, you can pick any word you'd like from your list and say it's the word that you had chosen all along. The beauty of this setup is that your opponent will have no way of knowing that you were dodging guesses the whole time – it looks like you simply picked an unusual word and stuck with it the whole way.

The Evil Hangman Project

Your final project is to write a computer program which plays a game of *Hangman* using this "Evil Hangman" algorithm. In particular, your program should do the following:

1. Read the file dictionary.txt, which contains the full contents of the *Official Scrabble Player's Dictionary, Second Edition*. This word list has over 120,000 words, which should be more than enough for our purposes. You should group the words based on their length for easy future use.
2. Prompt the user for a word length, reprompting as necessary until she enters a number such that there's at least one word that's exactly that long. That is, if the user wants to play with words of length -42 or 137, since no English words are that long, you should reprompt her.
3. Prompt the user for a number of guesses, which must be an integer greater than zero. Don't worry about unusually large numbers of guesses – after all, having more than 26 guesses is clearly not going to help your opponent!
4. Prompt the user for whether she wants to have a running total of the number of words remaining in the word list. This completely ruins the illusion of a fair game that you'll be cultivating, but it's quite useful for testing (and grading!) Please make sure that you have this step as we will be grading based on if you running total of words remaining matches the solution.
5. Play a game of *Hangman* using the Evil Hangman algorithm, as described below:
    a. Construct a list of all words in the English language whose length matches the input length.
    b. Print out how many guesses the user has remaining, along with any letters the player has guessed and the current blanked-out version of the word. If the user chose earlier to see the number of words remaining, print that out too.
    c. Prompt the user for a single letter guess, reprompting until the user enters a letter that she hasn't guessed yet. Make sure that the input is exactly one character long and that it's a letter of the alphabet.
    d. Partition the words in the dictionary into groups by word family.
    e. Find the most common "word family" in the remaining words, remove all words from the word list that aren't in that family, and report the position of the letters (if any) to the user. If the word family doesn't contain any copies of the letter, subtract a remaining guess from the user. If the game is not over, go back to step b.
    f. If the player has run out of guesses, pick a word (the smallest by alphabetical order) from the word list and display it as the word that the computer initially "chose."
    g. If the player correctly guesses the word, congratulate her.
6. Ask if the user wants to play again and if answer is yes, go back to step 2. Remember to clean up all dynamically allocated memory space that is no longer needed for a new round of play. Otherwise, when the user plays many many rounds, you will eventually exhaust all memory.

Tips

1. *Letter position matters just as much as letter frequency*. When computing word families, it's not enough to count the number of times a particular letter appears in a word; you also have to consider their positions. For example, "BEER" and "HERE" are in two different families even though they both have two E's in them. Consequently, representing word families as numbers representing the frequency of the letter in the word will get you into trouble.
2. *Watch out for gaps in the dictionary*. When the user specifies a word length, you will need to check that there are indeed words of that length in the dictionary. You might initially assume that if the requested word length is less than the length of the longest word in the dictionary,

there must be some word of that length. Unfortunately, the dictionary contains a few "gaps." The longest word in the dictionary has length 29, but there are no words of length 27 or 26. Be sure to take this into account when checking if a word length is valid.

3. *Don't explicitly enumerate word families*. If you are working with a word of length *n*, then there are 2^n possible word families for each letter. However, most of these families don't actually appear in the English language. For example, no English words contain three consecutive U's, and no word matches the pattern E-EE-EE--E. Rather than explicitly generating every word family whenever the user enters a guess, see if you can generate word families only for words that actually appear in the word list. One way to do it is to use code from homework 11, for each word in the word list, compute its pattern, and count this pattern. After enumerating through all the words in the word list, you can find the pattern with the most count.
4. *Tie breakers.* If you have multiple patterns with the same number of words associated with each pattern and they are all the largest sets of word families, how would you choose which family to go with next? In general, it is better to choose the word family that implies the user had wrong guess instead of right guess. And since '_' appears before all lower case letters 'a'-'z' in the alphabet in ascii table, pattern for a wrong guess, where all '_' remains will appear alphabetically before a pattern for a correct guess. So choosing the word family with the smallest pattern is the right choice for breaking a tie when multiple patterns have the same largest size.

Sample Output

```
What is the name of the dictionary file?
dictionary.txt
total words in the dictionary: 127142
How many letters do you want your word to have?
2
How many guesses do you want to have?
10
Do you want to know the running total of remaining words?(Y(y)/N(n)
y
Number of guesses remaining: 10
Number of words remaining: 94
Letters that has been guessed:

__
What letter?
a
Sorry, a is not in the word.
Number of guesses remaining: 9
Number of words remaining: 68
Letters that has been guessed: a

__
What letter?
e
Sorry, e is not in the word.
Number of guesses remaining: 8
Number of words remaining: 49
Letters that has been guessed: a e

__
What letter?
```

```
i
Sorry, i is not in the word.
Number of guesses remaining: 7
Number of words remaining: 36
Letters that has been guessed: a e i
__
What letter?
o
Sorry, o is not in the word.
Number of guesses remaining: 6
Number of words remaining: 14
Letters that has been guessed: a e i o
__
What letter?
u
Number of guesses remaining: 6
Number of words remaining: 6
Letters that has been guessed: a e i o u
u_
What letter?
h
Sorry, h is not in the word.
Number of guesses remaining: 5
Number of words remaining: 5
Letters that has been guessed: a e i o u h
u_
What letter?
s
Sorry, s is not in the word.
Number of guesses remaining: 4
Number of words remaining: 4
Letters that has been guessed: a e i o u h s
u_
What letter?
p
Sorry, p is not in the word.
Number of guesses remaining: 3
Number of words remaining: 3
Letters that has been guessed: a e i o u h s p
u_
What letter?
m
Sorry, m is not in the word.
Number of guesses remaining: 2
Number of words remaining: 2
Letters that has been guessed: a e i o u h s p m
u_
What letter?
```

n
Sorry, n is not in the word.
Number of guesses remaining: 1
Number of words remaining: 1
Letters that has been guessed: a e i o u h s p m n
u_
What letter?
t
Good job, you guessed the word: ut
Do you want to play again? Y(y)
y
How many letters do you want your word to have?
3
How many guesses do you want to have?
20
Do you want to know the running total of remaining words?(Y(y)/N(n)
y
Number of guesses remaining: 20
Number of words remaining: 962
Letters that has been guessed:

___
What letter?
a
Sorry, a is not in the word.
Number of guesses remaining: 19
Number of words remaining: 666
Letters that has been guessed: a

___
What letter?
e
Sorry, e is not in the word.
Number of guesses remaining: 18
Number of words remaining: 452
Letters that has been guessed: a e

___
What letter?
i
Sorry, i is not in the word.
Number of guesses remaining: 17
Number of words remaining: 311
Letters that has been guessed: a e i

___
What letter?
o
Number of guesses remaining: 17
Number of words remaining: 140
Letters that has been guessed: a e i o
_o_

What letter?
t
Sorry, t is not in the word.
Number of guesses remaining: 16
Number of words remaining: 118
Letters that has been guessed: a e i o t
_o_
What letter?
b
Sorry, b is not in the word.
Number of guesses remaining: 15
Number of words remaining: 98
Letters that has been guessed: a e i o t b
_o_
What letter?
m
Sorry, m is not in the word.
Number of guesses remaining: 14
Number of words remaining: 83
Letters that has been guessed: a e i o t b m
_o_
What letter?
n
Sorry, n is not in the word.
Number of guesses remaining: 13
Number of words remaining: 70
Letters that has been guessed: a e i o t b m n
_o_
What letter?
r
Sorry, r is not in the word.
Number of guesses remaining: 12
Number of words remaining: 62
Letters that has been guessed: a e i o t b m n r
_o_
What letter?
s
Sorry, s is not in the word.
Number of guesses remaining: 11
Number of words remaining: 50
Letters that has been guessed: a e i o t b m n r s
_o_
What letter?
c
Sorry, c is not in the word.
Number of guesses remaining: 10

Number of words remaining: 41
Letters that has been guessed: a e i o t b m n r s c
_o_
What letter?
p
Sorry, p is not in the word.
Number of guesses remaining: 9
Number of words remaining: 30
Letters that has been guessed: a e i o t b m n r s c p
_o_
What letter?
w
Sorry, w is not in the word.
Number of guesses remaining: 8
Number of words remaining: 21
Letters that has been guessed: a e i o t b m n r s c p w
_o_
What letter?
v
Sorry, v is not in the word.
Number of guesses remaining: 7
Number of words remaining: 20
Letters that has been guessed: a e i o t b m n r s c p w v
_o_
What letter?
l
Sorry, l is not in the word.
Number of guesses remaining: 6
Number of words remaining: 17
Letters that has been guessed: a e i o t b m n r s c p w v l
_o_
What letter?
u
Sorry, u is not in the word.
Number of guesses remaining: 5
Number of words remaining: 15
Letters that has been guessed: a e i o t b m n r s c p w v l u
_o_
What letter?
y
Sorry, y is not in the word.
Number of guesses remaining: 4
Number of words remaining: 9
Letters that has been guessed: a e i o t b m n r s c p w v l u y
_o_
What letter?
h
Sorry, h is not in the word.

```
Number of guesses remaining: 3
Number of words remaining: 6
Letters that has been guessed: a e i o t b m n r s c p w v l u y h
_o_
What letter?
j
Sorry, j is not in the word.
Number of guesses remaining: 2
Number of words remaining: 5
Letters that has been guessed: a e i o t b m n r s c p w v l u y h j
_o_
What letter?
f
Sorry, f is not in the word.
Number of guesses remaining: 1
Number of words remaining: 3
Letters that has been guessed: a e i o t b m n r s c p w v l u y h j f
_o_
What letter?
r
Not a valid guess, try again.
w
Not a valid guess, try again.
g
Number of guesses remaining: 1
Number of words remaining: 2
Letters that has been guessed: a e i o t b m n r s c p w v l u y h j f g
go_
What letter?
z
Sorry, z is not in the word.
Sorry, you run out of guesses, and the word is: god

Do you want to play again? Y(y)
n
Press any key to continue . . .
```

How to submit

Final Project's solution should be all included in one "evilhangman.cpp" file. This is the file you are submitting to ANGEL's Final Project Dropbox as an ATTACHMENT. The final project is due 4/26/2012 11:59:59pm. Notice that Final Project Dropbox will be in **Overflow 1: CMPSC 121 – SP12 on Angel**.

Acknowledgement

This project is adapted from Keith Schwarz (htiek@cs.stanford.edu)'s nifty assignment - Evil Hangman.