For this assignment you will be writing a program to cheat at the game Hangman.

# Hangman Rules

The game hangman is a word guessing game. A word is selected and a hint is constructed from the word by replacing all non-guessed letters with underscores. At each turn the player guesses a letter. If that letter is present in the word, the letters that match it in the word are revealed. If not, then the number of incorrect guesses increases.

A player may only guess a single letter at a time, and can not repeat a guess.

The game ends by two conditions. First, when the number of incorrect guesses is at least 5, the player loses. Second, when the player has guessed all of the letters in the word, in which case the player wins.

# The twist

While you will be writing a program to play Hangman, we want to make it cheat as much as possible, while still presenting only correct information. Instead of selecting a word to start, we will have a list of words. Whenever the player guesses a word, the list of remaining words is partitioned based upon the information that would be revealed about the word. Then the partition with the maximum size is selected as our new word list. By doing this we ensure that at each step there is a maximal amount of possible words that the player needs to guess from.

# Your Program

Your program to play the game should read a filename from the command line. If there is an additional argument, regardless of what it is, it should print the full details about what is happening behind the scenes, otherwise it should not.

```
$ ./hangman wordlist.txt
```

When called like this, the program should read the file `wordlist.txt` as the source of words and play the game without showing any additional information beyond what it necessary to play the game. That is don't print out the internal word lists or partitions.

```
$ ./hangman wordlist.txt foo
```

When called like this, the program should read the file `wordlist.txt` as the source of words and play the game showing the full details. The sample output below would be an example of this type of output.

Your program should be compiled with CMake following the **Project setup for CMake** directions. It also needs to be split into various files at logical boundaries.

# The code

There is substantial starting code for this assignment that you need to use. There are three fundamental data structures used. A string, a dynamically growing array, and an associative array. In a language like Python these would be the built-in string, a list and dictionary, respectively. However C does not provide these. Strings exist as null terminated character arrays. But the others do not exist as part of the standard library and will be provided.

You should use as a source of words, a word list. This should be a file that simply contains a long list of words. If you search around for word lists, they are not hard to find, for example one can be **here (https://github.com/dwyl/english-words/blob/master/words_alpha.txt)** . If you are on Linux or MacOS, you can find a list at /usr/share/dict/words. When testing, you probably do not want to use the whole word list as it will be very difficult to verify correct behavior. Instead create words lists of your own that contain the patterns you are looking for, don't concern yourself with them being actual words.

The general flow of your program should be:

1. Read in the dictionary file and filter for a specific word length
2. Prompt the user for a guess
3. Partition the word list based upon the guessed letters so far, revealing the mask that has the most ambiguity.
4. Prompt the user for another guess and repeat.

Specific components you will need:

## Masking a word

```
void mask(char *word, char *guesses, char *output);
```

This function should take in a word and a string representing the guesses from the player. It will copy character by character from the word to the output so that the letter is preserved if present in the guesses string, and replaced with an underscore if it is not.

For example, if you have the word zymurgy and the letters guessed are "rstne", its masked form would be ____r__. If the letters guessed were "ym", its masked form would be _ym___y.

# Partitioning

If you start with the word list {abcd, abce, abdg}, have already guessed ab, leading to the revealed ab__ and then guess c, the resulting partitions should be abc_ : {abcd, abce} and ab__ : {abdg}. The decision should then be to reveal abc_ and make the word list become {abcd, abce}. Note that to collect all the words for a given mask together is where the associative array data structure comes into play.

# Game Loop

Generally any game has a game loop to control playing the game. The general procedure for our game is as follows:

```
setup
while game is not over
    print current state of game
    read user input
    apply user input
print win/loss message
```

In particular for this game what needs to occur:

Setup: Read in the dictionary file. Setup the hint as all underscores

Game Over Check: The game is over if the current hint no longer has any underscores or if the user has run out of guesses

Print current state of the game: Print out the current hint, guesses made, etc

Read user input: Prompt the user for their letter selection, verifying that they haven't already guessed it

Apply user input: Detailed below

## Applying user input

When cheating, the procedure to cheat is pretty straight forward

```
for each word in the word list
    compute the masked version of the word
    insert the word into the red-black tree with key of the masked version and the value the word
Find the key that corresponds to the largest list of words
That key will be the new hint displayed to the user
Delete the existing word list and replace it with the one from the partition
Decide whether a guess was "correct" based upon whether the hint changed
```

# Suggested Approach

Build out the game to play honestly.

1. Write the main game loop, using just the first index of the read in words initially
2. Write the display information for normal gameplay
3. Write the code to reader user input
4. Write the function to mask a word, use it to apply a move
5. Write the game over checks
6. Write win/loss messages at the game end

After this point, make it cheat.

1. Incorporate the provided red-black tree
2. Swap out the apply move code for partitioning code
3. Write the displaying of the internals if the command line specifies it
4. Update the win/loss messages

# Starting Code

There are two main data structures:

## WordList

In **WordList.h** and **WordList.c** s a dynamically growing array. This is used in two main ways. First it used to store the words when read in from the word list. Second it is used as the value type in the associative array

# WordPartitions

Will be posted on May 27. This is a red-black tree that will work as an associative array. It is used during the partitioning step when cheating.

# Sample Output

User input is in bold

```
What word length do you want? (<1 or non-numeric initiates self-tests) 5
There are 43 words of length 5
----------------------------------------
The possible words are: aping awash based beers beige bided blued bossy chose cubit dirks fains f
alse fence fishy fours gowns inked khaki knoll knows mains moody noise pepsi serfs shied skits sn
owy spent sunny swirl taffy timid twerp until viler wayne wheel where writs yelps zeros
There are 43 words still possible.
You have 5 incorrect guesses left.
Current hint: _____
Guessed letters:
What letter do you want to guess? a
You have guessed 'a'
The partite sets from your guess are:
_____: beers, beige, bided, blued, bossy, chose, cubit, dirks, fence, fishy, fours, gowns, inked,
knoll, knows, moody, noise, pepsi, serfs, shied, skits, snowy, spent, sunny, swirl, timid, twerp,
until, viler, wheel, where, writs, yelps, zeros,
__a__: khaki,
_a___: based, fains, false, mains, taffy, wayne,
a____: aping,
a_a__: awash,
Unfortunately 'a' is not in the word
----------------------------------------
The possible words are: beers beige bided blued bossy chose cubit dirks fence fishy fours gowns i
nked knoll knows moody noise pepsi serfs shied skits snowy spent sunny swirl timid twerp until vi
ler wheel where writs yelps zeros
There are 34 words still possible.
You have 4 incorrect guesses left.
Current hint: _____
Guessed letters: a
What letter do you want to guess? e
You have guessed 'e'
The partite sets from your guess are:
_____: bossy, cubit, dirks, fishy, fours, gowns, knoll, knows, moody, skits, snowy, sunny, swirl,
timid, until, writs,
____e: chose, noise,
___e_: bided, blued, inked, shied, viler,
__e__: spent, twerp,
__e_e: where,
__ee_: wheel,
_e___: pepsi, serfs, yelps, zeros,
_e__e: beige, fence,
_ee__: beers,
Unfortunately 'e' is not in the word
----------------------------------------
The possible words are: bossy cubit dirks fishy fours gowns knoll knows moody skits snowy sunny s
wirl timid until writs
There are 16 words still possible
```

```
There are 10 words still possible.
You have 3 incorrect guesses left.
Current hint: _____
Guessed letters: ae
What letter do you want to guess? i
You have guessed 'i'
The partite sets from your guess are:
_____: bossy, fours, gowns, knoll, knows, moody, snowy, sunny,
___i_: cubit, until,
__i__: skits, swirl, writs,
_i___: dirks, fishy,
_i_i_: timid,
Unfortunately 'i' is not in the word
----------------------------------------
The possible words are: bossy fours gowns knoll knows moody snowy sunny
There are 8 words still possible.
You have 2 incorrect guesses left.
Current hint: _____
Guessed letters: aei
What letter do you want to guess? o
You have guessed 'o'
The partite sets from your guess are:
_____: sunny,
__o__: knoll, knows, snowy,
_o___: bossy, fours, gowns,
_oo__: moody,
You have guessed correctly!
----------------------------------------
The possible words are: knoll knows snowy
There are 3 words still possible.
You have 2 incorrect guesses left.
Current hint: __o__
Guessed letters: aeio
What letter do you want to guess? k
You have guessed 'k'
The partite sets from your guess are:
__o__: snowy,
k_o__: knoll, knows,
You have guessed correctly!
----------------------------------------
The possible words are: knoll knows
There are 2 words still possible.
You have 2 incorrect guesses left.
Current hint: k_o__
Guessed letters: aeiok
What letter do you want to guess? n
You have guessed 'n'
The partite sets from your guess are:
kno__: knoll, knows,
You have guessed correctly!
----------------------------------------
The possible words are: knoll knows
There are 2 words still possible.
You have 2 incorrect guesses left.
Current hint: kno__
Guessed letters: aeiokn
What letter do you want to guess? l
You have guessed 'l'
The partite sets from your guess are:
kno__: knows,
knoll: knoll,
Unfortunately 'l' is not in the word
```

```
-----------------------------------------
The possible words are: knows
There are 1 words still possible.
You have 1 incorrect guesses left.
Current hint: kno__
Guessed letters: aeioknl
What letter do you want to guess? w
You have guessed 'w'
The partite sets from your guess are:
know_: knows,
You have guessed correctly!
-----------------------------------------
The possible words are: knows
There are 1 words still possible.
You have 1 incorrect guesses left.
Current hint: know_
Guessed letters: aeioknlw
What letter do you want to guess? s
You have guessed 's'
The partite sets from your guess are:
knows: knows,
You have guessed correctly!
You win! The word was knows
```