**CS 351 - Advanced Data Structure Practicum** **Spring 2023**

# Project 2: Spelling Bee

**Due**: Wednesday, March 1st, 2023 @ 11:59PM

- 24 hours late: Thursday, March 2nd, 2023 @ 11:59pm for 10% deduction
- 48 hours late: Friday, March 3rd, 2-23 @ 11:59pm for 30% deduction (recall the midterm exam is scheduled for Friday March 3rd and will not change!)

Optional: You may work on this project with one other student currently taking CS 351. If you work on this project with a partner, you are to submit only one submission for the both of you. Gradescope will allow you to select your partner when submitting. It is your responsibility to properly do this.

---

**Objectives**:

1. Use a Trie data structure to store lists of words
2. Practice with tree-type structures
3. Play around a slight bit with inheritance

---

**Summary**: We are implementing a version of the Spelling Bee puzzle that the New York Times has as part of its daily games. To access (and play) the New York Times version clink: https://www.nytimes.com/puzzles/spelling-bee

The basics of the Spelling Bee game is that you are given one "central letter" and 6 "other letters". The player's job is to find as many words as possible that are:

- Four letters long or longer
- Contain the "central letter"
- May contain the "other letters"
- Allowed letters may be used more than once
- Words must exist in the current word dictionary
- The player can make as many guesses as they wish when finding possible words

As each word is found the game keeps track of the following information:

1. The player's total score:
   a. a four letter word is worth only 1 point
   b. other words are worth 1 point per letter
   c. Pangrams (word that contain all 7 letters) get 7 additional points
2. Whether the player has found a Pangram. A Pangram is a word that contains all of the 7 letters. Pangrams may contain a letter multiple times.
3. Whether the player has scored a Bingo. A Bingo is when the player has found words that starts with each of the 7 possible letters.

Your program is to create a C++ class named Trie. The purpose of using the Trie data structure is to store the dictionary and to allow for faster look-up of potential words. Your program is to create a C++ class SBTrie which is an inherited class from Trie.

Our code must be written in three source code files:
Trie.h, SBTrie.h and spellb.cpp

Starter code for these three files have been provided.

The SBTrie class adds on one primary additional method to the Tries class that is useful for the Spelling Bee game: retrieving a list of all words that satisfy the conditions of the game. (Trie has a method that retrieves all words, but this is way too inefficient to really be useful for the purposes of the game to retrieve all the words and then remove the words that don't satisfy the game conditions.)

The Trie class is to have the following operations/methods:

- a constructor
- a destructor
- bool getFromFile(string filename)
  - enter all of the words that exist from the file specified by the given filename. Words from the file are separated by whitespace characters and must only contain letters. Return a boolean to indicate success/failure. Using the insert() method is expected. (file not accessible/readable/etc.)
- bool insert(string word)
  - inserts the word given by the parameter into the trie. This function will allocate new nodes for the trie if needed. The function returns a boolean value of true if the word was successfully inserted into the trie. Fails if the word already exists, contains non-letters, or allocation errors occur.
- bool search (string word) const
  - returns true if the word given by the parameter exists in the trie. Fails otherwise
- bool remove (string word)
  - removes the word given by the parameter from the trie. Returns true of the word was successfully removed (false otherwise). Deallocates nodes from the trie only if those nodes no longer contain any words or parts of words for the trie.
- bool clear()
  - removes all words from the trie and deallocates all nodes. Returns true unless deallocation error occurs.
- int wordCount() const
  - returns the number of words currently stored in the trie. This operation is to have an O(1) runtime.
- str::vector<string>* words() const
  - builds and returns a vector of strings containing all of the words in the trie. The words are to be stored in the vector in sorted ascending order.

The SBTrie class must inherit from the Trie class and must have the following additional operation/method:

- str::vector<string>* sbWords(char centralLetter, string otherLetters) const
    - builds and returns a vector of strings containing all of the words in the trie that meet the criteria of the Spelling Bee game:
        - words must be at least 4 letters long
        - words must contain the centralLetter
        - in addition to the centralLetter, words may contain only the letters specified in the otherLetters string.
    - this method must not have an O(N) run time.
    - It needs to have a runtime of O(length of longest valid word)
- You are to use the SBTrie class to keep track of additional information for the Spelling Bee game. The exact data members and methods are for you to determine. Ideas for this could be:
    - central letter for Spelling Bee game
    - allowed letters for Spelling Bee game
    - trie of the current words found/discovered for the Spelling Bee game (use aninstance of the trie data structure to store the found/discovered words by the user which is different from the trie used to store the word dictionary)
    - current score for the Spelling Bee Game
    - Pangram-found status
    - Bingo-found status
- The methods of the SBTrie class and the Trie class MUST NOT to print out any information. All print outs for the program are to occur in the file of spellb.cpp. There are a number of functions included in the spellb.cpp file that correspond to each of the required commands for the program. These functions are the functions that are to print out the information required by the program. Any information that is printed out from the SBTrie class, Trie class, or the files containing them will result in a deduction of points.

---

The Spelling Bee game that we will implement with an interactive C++ program has the following command options:

1. Enter a new dictionary – The user is to unput the name of the file containing the dictionary after entering the command of 1. This command is to first clear out the existing word dictionary from the trie before processing the words contained in the file. A “word” is any whitespace delimited character sequence contained in the file. Words that contain upper case letters should be made lower case. Words that contain non-letter characters will not be stored in the trie.

2. Add to the existing dictionary – The user is to unput the name of the file containing the dictionary after entering the command of 2. This command will keep all the existing words in the trie dictionary. A “word” is any whitespace delimited character sequence contained in the file. Words that contain upper case letters should be made lower case. Words that contain non-letter characters will not be stored in the trie.

3. Enter a new set of 7 letters that make up the “central letter” and the “other letters” for the Spelling Bee game – These letters must be entered without any

whitespace characters between any of the letters. The first character entered is the “central letter”. The next 6 letters become the “other letters”.

a. The game will use lower case letters. If the user enters an upper case letter, convert it to the same letter in lower case.
b. If the user enters a character that is not a letter, ignore that character when processing the new set of letters. (Thus, the user input of “a-bcdefg” or “a,b,c,d,e,f,g” or “abcdefg” would all be valid.)
c. The user must enter 7 letters in the set and each letter in the set must be unique. If the user does not enter 7 unique letters, display the error message of “Invalid letter set”.
d. Once the program has determined that the user has entered a valid set of 7 letters: clear out the trie that contains the found words, reset the Spelling Bee score to zero, set the Pangram-found indicator to false, and set the Bingo-found indicator to false.

4. Display the current “central letter” and the 6 “other letters” in the following manner:

   **Central Letter:  a**
   **6 Other Letters: b,c,d,e,f,g**

5. Enter a potential word – The word entered must be checked that it is a valid Spelling Bee word before it is entered into the trie containing the found words. Recall the valid word must be at least 4 characters long, must contain the central letter, can only contain the central letter and the 6 other letters, must exist in the dictionary trie, and has not yet been found by the player. If the player enters a character that is not a letter, treat it as an invalid letter. If an invalid word is entered by the player report one of the following error message. If the invalid word is invalid for multiple reason, report the first message from the following list:

   **word is too short**
   **word is missing central letter**
   **word contains invalid letter**
   **word is not in the dictionary**
   **word has already been found**

   If the word is a new valid word, add it into the trie containing the found words and print out a message showing the word found, the point value of the word, the total point values scored, whether the word is a Pangram, and if this word scores a Bingo for the player. The following would be valid messages for words for the Central Letter of f and other letter of abeilx:

   **found able 1 point, total 1 point**
   **found ball 1 point, total 2 points**
   **found exile 5 points, total 7 points**
   **found fixable 14 points, total 21 points, Pangram found**

   When the seventh word of a Bingo is found add “**, Bingo scored**” to the output. (This example does not have a word that begins with x so no Bingo is possible.) If the seventh word of a Bingo is also a Pangram, report the Pangram before reporting the Bingo (i.e.: **..., Pangram found, Bingo scored**)

6. Display all words found and other information – The words are to be displayed in alphabetical order with one word on a line. On the line following the last word found, display total number of words found, the total points scored, and whether the user has found a Pangram or scored a Bingo. This last line should appear as follows (assuming both Pangram and Bingo):

   **10 words found, total 54 points, Pangram found, Bingo scored**

7. List all possible Spelling Bee words from the dictionary – The words are to be displayed in alphabetical order with one word on a line followed by the length of the word and if the word is a Pangram. Align the numbers in a right justified column at column 20 (if a word is longer than 17 characters, print a space after the word before the length – std::setw(x), std::right and std::left from the C++ iomanip library can help do this). The output should appear as follows:
```
12345678901234567890
affable            7
affix              5
alfalfa            7
...
fill               4
fixable            7 Pangram
flab               4
...
```

8. Display the list of commands for the assignment – This code should already be given in the starter code.

9. Quit the program – This code should already be given in the starter code.

## Additional Rules and Hints:

**RULE**: You may use the vector and the string class from the STL (although you don't *have* to). All other external data structure classes (not written by you) are forbidden. The trie node must NOT contain a string showing the current sequence of letters needed to get to that node; however, the node may have a single character showing the most recent letter used to traverse to that node.

**RULE**: None of your methods should be printing anything to the terminal!!! If you inject print statements for debugging purposes, be sure to remove them before final submission! Expect penalties for extraneous output! You might consider using a trick of a including a “Debug Mode” private data member.

**HINTS/SUGGESTIONS:** On pencil and paper, start with a straightforward implementation that does not necessarily meet the runtime requirements (but meets the behavioral requirements). Analyse the runtime of the various operations. Now start thinking of alternative ways to organize the data to meet runtime requirements.)

Come to class! We will be brainstorming strategies!

## Testing
You need to test your own implementation! Be creative and try to break it!

## Submission
Your primary deliverable is four files: spellb.cpp, SBTrie.h, Tries.h and a makefile. Submission will be via Gradescope.

**Sample Sets of Spelling Bee Letters from the New York Times:**

```
Central Letter: C     Other Letters: I,K,L,P,R,Y
Central Letter: I     Other Letters: A,C,H,N,T,Y
Central Letter: P     Other Letters: C,E,M,N,O,T
Central Letter: P     Other Letters: Y,A,E,T,C,H
Central Letter: F     Other Letters: I,X,A,B,L,E
Central Letter: D     Other Letters: A,C,I,N,O,R
Central Letter: G     Other Letters: A,C,I,K,M,N
```