# KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

# Information and Computer Science Department

# ICS-202 Data Structures

## Lab Project

Guidelines:

1. Due date for the project is 1st May 2020 (Friday).

2. This project is strictly individual.

3. There will be a project demonstration followed by Question/Answer Session from the instructor online. Details will be announced later.

**PROJECT QUESTION: PART 1**

Write a java class that implements a **Trie** of uppercase alphabetical letters. A trie is a data structure that permits efficient lookup of words in a dictionary (See figure). You may need to make two classes i.e. **TrieNode** and **Trie**. Your **trie** should support the following operations:

(a) contains(String s) [return type boolean] = true if the word ‘s’ is contained in the trie.

(b) isPrefix(String p) [return type boolean] = true if ‘p’ is a prefix of any word in the trie.

(c) insert(String s) [return type void] = inserts the word ‘s’ in the trie.

(d) delete(String s) [return type void] = deletes ‘s’ from the trie.

(e) isEmpty() [return type boolean]

(f) clear() [return type void]

(g) constructors, accessors, mutators = as needed

(h) allWordsPrefix(String p) [return type String[]] = this method returns all words whose prefix is ‘p’.

(i) size() [return type int] = returns the number of nodes in the trie.

Initially your program will take from the user a list of letters (or a word). It will then make all arrangements of the letters in this word and insert them in the trie if these are valid words from the dictionary. This procedure is to be done only the first time (initially).

Now write a main program to test your trie. A sample session is given as follows:

Enter your list of letters> S T O P

Trie Created.

For the following operations, you can use a menu.

Enter a word to search for> STOP

Word Found: STOP

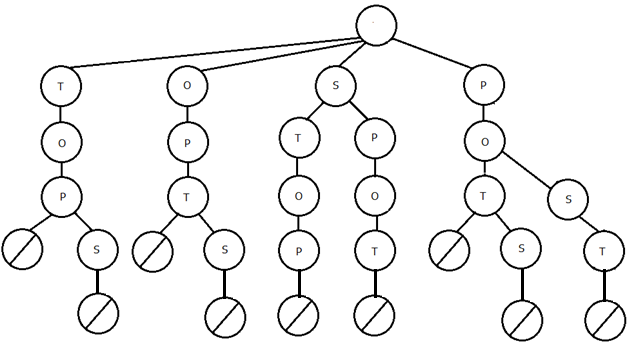
Enter a prefix> PO

Found the following words: POT, POTS, POST

Enter a word to insert> …

Enter a word to delete> …

then based on the dictionary, the following trie is created:



For this project, a dictionary file is included.

**PART II: PROJECT INTERFACE**

Menu driven: Your project should have a menu in the beginning. This is to ensure that your work can be tested by inserting, deleting and listing all words/prefixes. A possible menu is shown as follows:

TRIE PROJECT: Enter your choice?

1) Create an empty trie

2) Create a trie with initial letters

3) Insert a word

4) Delete a word

5) List all words that begin with a prefix

6) Size of the trie

7) End

A description for some of the operations is given as follows:

1) **Create an empty trie**: This will create an empty trie with no words at all.

2) **Create a trie with initial letters**: This will prompt the user for initial list of letters.

Then it will generate all permutations of these letters and insert valid words in the trie.

Here it has to be mentioned that the permutations can also be of shorter length than the

list of letters and that no letter is to be repeated. For example, if the initial list of

letters is "STOP", then "TOP" and "TOPS" are both valid permutations, but "STOPS" isn't.

3) **Insert a word**: This inserts a word into the trie.

4) **Delete a word**: This deletes a word from the trie. Suppose that your trie contains the words

"TOP" and "TOPS", a deletion of "TOPS" will still retain the word "TOP" in the trie.

5) **List** all words that begin with a prefix

6) **Size of the trie**: This should count all nodes that have a valid letter in them. For example, the

size of a trie with the words "TOP" and "TOPS" is 4.

7) **End**