UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

# Department of Computer Science

## COS212: Practical 6

Release: Monday 24 May 2021, 18:00
Deadline: Friday 28 May 2021, 18:00

# PLAGIARISM POLICY

## UNIVERSITY OF PRETORIA

The Department of Computer Science considers plagiarism as a serious offence. Disciplinary action will be taken against students who commit plagiarism. Plagiarism includes copying someone else's work without consent, copying a friend's work (even with consent) and copying material (such as text or program code) from the Internet. Copying will not be tolerated in this course. For a formal definition of plagiarism, the student is referred to `http://www.library.up.ac.za/plagiarism/index.htm` (from the main page of the University of Pretoria site, follow the *Library* quick link, and then choose the *Plagiarism* option under the *Services* menu). If you have any form of question regarding this, please ask one of the lecturers, to avoid any misunderstanding. Also note that the OOP principle of code re-use does not mean that you should copy and adapt code to suit your solution.

# Objectives

The aim of this practical is to learn how Tries work and how to implement this functionality.

# Instructions

Complete the tasks below. Certain classes have been provided for you in the *files* zip archive of the practical. You have also been given a main file which will test some code functionality, but it is by no means intended to provide extensive test coverage. You are encouraged to edit this file and test your code more thoroughly. Remember to test boundary cases. Upload **only** the given source files with your changes in a compressed archive before the deadline. Please comment your name **and** student number in at the top of each file.

# Tries

Tries are trees that use parts of the key to guide navigation within the tree. Every node in a trie contains either an entire key (in which case it is a leaf node) or it contains a set of pointers to subtries (in which case it is a non-leaf node). For simplicity's sake, suppose that a trie only needs to be able to store keys consisting of the characters {`Q,W,E,R,T,Y`}. If the pointers array of a node at level $i$ has en entry at index $k$, then the $i_{th}$ letter in the key being processed is the $k_{th}$ letter in {`#,E,Q,R,T,W,Y`}. Note that the character '#' is used to represent the end of a key. For further explanation of how tries work, refer to page 377 of the textbook.

For this practical, you are required to implement a trie. The trie must support keys that consist of characters from a fixed charset (like {`Q,W,E,R,T,Y`} in the example above). You do not need to perform any compression on the trie. You have been provided with the basic structure and some helper functions for the `Trie` class. First a brief explanation of the functions that have been already implemented:

`Trie(char[] letters)`
: Initializes the trie structure. The given set of letters are stored so that the order of pointers in the nodes is known. Notice that the end-of-word character '#' should not be specified in the input array; it will be added automatically. The $0_{th}$ index of a `ptrs` array will correspond to the end-of-word character.

`int index(char c)`
: This is a helper function that may be useful for interaction with `ptrs` arrays. This function will return the index in a `ptrs` array that corresponds to the character `c` .

`char character(int i)`
: This is a helper function that may be useful for interaction with `ptrs` arrays. This function will return the character corresponding to an index $i$ in a `ptrs` array.

`String nodeToString(Node node)`
: Returns a string representation of the given node. If the node is a leaf, then the result is

just its key. If the node is a non-leaf, then this returns a set of pairs, such as '(#,1) (E,1) (Q,0) (R,0) (T,1) (W,1) (Y,1)'. This indicates that the pointers corresponding to the characters '#', 'E', 'T', 'W' and 'Y' are not null. Also, that the pointers corresponding to the characters 'Q' and 'R' are null.

`print()`
: This prints the tree by means of a breadth-first traversal.

You have also been provided with a `Node` class which defines the nodes of the trie. You may not edit this class. The class has two constructors, one for a leaf node instance and one for a non-leaf node instance. Observe that the `ptrs` array of leaf nodes will be empty and that the `key` field of non-leaf nodes will be empty.

Your task is to implement the following methods in the `Trie` class according to the given specification.

# Task 1: Insert Key [40]

`public void insert(String key)`
: This function should insert the given key *key* into the trie at the correct position. You need to create the necessary non-leaf nodes to structure it correctly.

# Task 2: Contains Key [5]

`public boolean contains(String key)`
: This function should determine if a node with the given key *key* exists. If the node exist, return `true`, otherwise return `false`.

# Task 3: Print Keys [5]

`public void printKeyList()`
: This function should print all the keys in the trie in alphabetical order. The printed list should be terminated by a newline char. You may assume the charset used by the trie is in alphabetical order.

You may notice the presence of a `Queue` class, which is used for breadth-first traversal in the trie's `print` function. You may not edit this class. However, you may use this class in your implementations, though keep in mind that the queue is First-In-First-Out. You can use both iterative and recursive approaches in your implementations

# Submission

You need to submit your source files on the Assignment website `https://ff.cs.up.ac.za/`. All tasks need to be implemented (or at least stubbed) before submission. Place **ONLY Tries.java** file in a zip or tar/gzip archive (you need to compress your tar archive) named uXXXXXXXX.zip or uXXXXXXXX.tar.gz where XXXXXXXX is your student number. You have 4 days to complete this practical, regardless of which practical session you attend. Upload your archive to the *Practical 6* slot on the Assignment website. Submit your work before the deadline. No late submissions will be accepted.