Module Code
COM00016C

BEng, BSc, MEng and MMath Degree Examinations 2020-2021

Department Computer Science

Title Software 2 - Practical Formative

Time Allowed 24 Hours (NOTE: As it is a formative assessment, and you have other module running at the same time, you will effectively have more than 24 hours).

Time Recommended ONE hour and THIRTY minutes

Word Limit Not Applicable

Allocation of Marks:

Question 1 is worth 100 marks, the code must adhere to the Google Java Style Guide.

Instructions:

This is a formative assessment and therefore the submission is **voluntary** and does not go towards your final mark.

Candidates should answer the question using Java 8 or above. Failing to do so will result in a mark of 0%. Download the paper and the required source files from the VLE, in the "Assessment>Practical Formative" section. Once downloaded, unzip the file. You **must** save all your code in the **ClosedExamination** folder provided. **Do not** save your code anywhere else other than this folder.

Submit your answers to the Department's Teaching Portal as a single **.zip** file containing the **ClosedExamination** folder and its sub-directories. Failing to include the **ClosedExamination** folder will result in a loss of 10 marks. Other archival format such as .rar and .tar files will not be accepted and will result in a mark of 0%.

If a question is unclear, answer the question as best you can, and note the assumptions you have made to allow you to proceed. Please inform ⟨`cs-exams@york.ac.uk`⟩ about any suspected errors on the paper immediately after you submit.

A Note on Academic Integrity

We are treating this online examination as a time-limited open assessment, and you are therefore permitted to refer to written and online materials to aid you in your answers.

However, you must ensure that the work you submit is entirely your own, and for the whole time the assessment is live you must not:

- communicate with departmental staff on the topic of the assessment
- communicate with other students on the topic of this assessment
- seek assistance with the assignment from the academic and/or disability support services, such as the Writing and Language Skills Centre, Maths Skills Centre and/or Disability Services. (The only exception to this will be for those students who have been recommended an exam support worker in a Student Support Plan. If this applies to you, you are advised to contact Disability Services as soon as possible to discuss the necessary arrangements.)
- seek advice or contribution from any third party, including proofreaders, online fora, friends, or family members.

We expect, and trust, that all our students will seek to maintain the integrity of the assessment, and of their award, through ensuring that these instructions are strictly followed. Failure to adhere to these requirements will be considered a breach of the Academic Misconduct regulations, where the offences of plagiarism, breach/cheating, collusion and commissioning are relevant: see AM1.2.1 *(Note this supercedes Section 7.3 of the Guide to Assessment)*.

1 (100 marks) t9 predictive text technology

T9, which stands for Text on 9 keys, was a predictive text technology used in the late 1990s for mobile phones, specifically those that contain a 3×4 numeric keypad as shown in Figure 1. T9's objective was to make it easier to type text messages. It allowed words to be entered by a single key press for each letter, as opposed to the multi-tap approach used in conventional mobile phone text entry at the time, in which several letters were associated with each key, and selecting one letter often required multiple key press. For example, to type the word *the*, the user had to type 3 keys *8-4-3* with predictive text technology as opposed to five key strokes *8-4-4-3-3* for "multi-tap" technology.

In ideal predictive text entry, all words used are in a given dictionary, punctuation are ignored, no spelling mistakes are made, and no typing mistakes are made. The user presses the number corresponding to each letter and, as long as the word exists in the predictive text dictionary it will appear. For instance, pressing *4-6-6-3* will typically be interpreted as the word *good*, provided that a linguistic database in English is currently in use, though alternatives such as *home*, *hood* and *hoof* are also valid interpretations of the sequence of key strokes.

The aim of this question is to retrieve all valid words (that is in a given dictionary) corresponding to a sequence of key strokes from a 3×4 numeric keypad. For simplicity, we assume all characters are lower case.

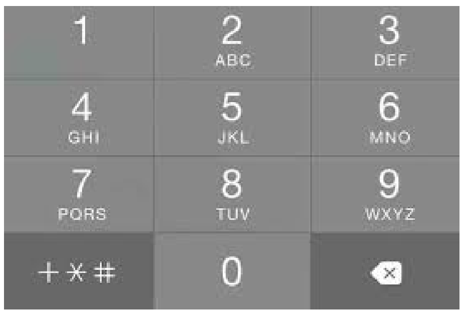

Figure 1: Example of a 3×4 numeric keypad used for T9.

(i) [25 marks] For this question, you must use the file `T9Pad.java` provided. The class `T9Pad` represents a numeric keypad. The mapping between alphabet characters and number is stored in the attribute `pad`. The attribute `pad` is a `HashMap` where keys are `Integer` and values are `Set<Character>`.

Implement `public Integer getKeyCode(Character letter)` which returns the digit associated with the character `letter`. If there are no mapping for this character, the method must return an `IllegalArgumentException`.
For example, assuming we only added the mapping `(2, "abc")` via the method `addKey`, `getKeyCode('a')` should return 2, whereas `getKeyCode('d')` must throw an exception.

(ii) [25 marks] Implement `public List<Character> getPadLetters()` in the class `T9Pad` which returns all the alphabet characters currently represented in the pad. The method should return an empty list if there are no mapping between numeric values and characters.

(iii) [25 marks] Words produced by the same combination of keypresses have been called "textonyms". For example, the key sequence 4663 on a keypad, correspond to the words *good* as well as other words, such as *home*, *gone* and so on. the words *good*, *home* and *gone* are "textonyms".

Implement `public boolean isTextonym(String word1, String word2)` in the class `T9Pad` which returns `true` if `word1` and `word2` are textonyms, `false` otherwise.

(iv) [15 marks] For the reminder of the question, complete the implementation of the class `T9Tree`. This class represents the dictionary of all words known by the user. In order to have an efficient predictive text application, we have decided to represent the data structure as a tree (see Figure 2). Each node represents the set of words that can be type using the numeric sequence from the root to that node. For example, to type the word *go*, we have to press the keys 4 then 6. Starting from the root, we must go to the child 6, then child 4 of child 6. For longer words such as *home* we keep going down the tree branches. In addition, the root cannot have any words, hence an empty set of words for the root.

Implement the method `public Set<String> getAllWords()` which returns all the known word in that tree.

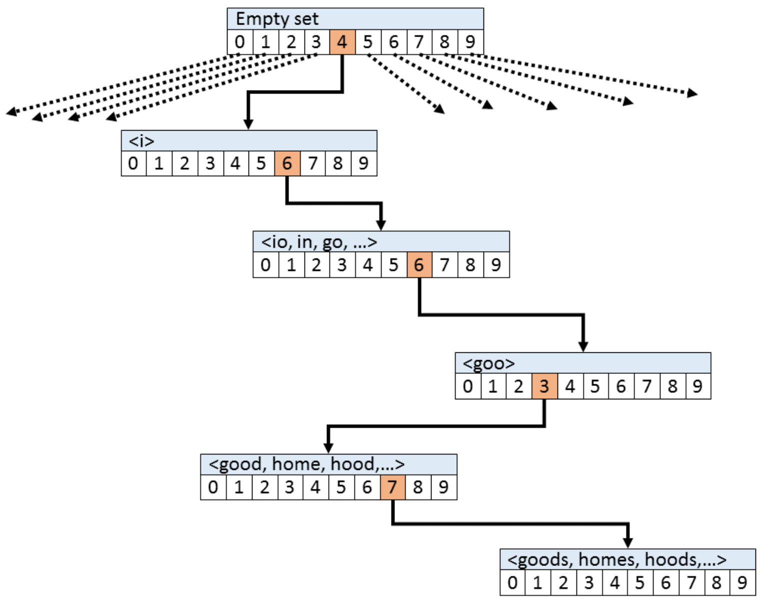

Figure 2: Tree structure to implement T9 predictive text. Each node can have up to 10 children.

(v) [10 marks] Implement `public Set<String> getAllWords(String t9code)` which returns all the known words in that tree such as their numeric code has the prefix `t9code`. For example `getAllWords("4663")` should returns a set containing `"good"`, `"goods"`, `"goodies"`, `"home"`, `"homes"` to list a few. If no known word have such a prefix, the method returns an empty set. The method throws an `IllegalArgumentException` if the `t9code` contains keys that are not part of the numeric pad.

Note, if `t9code` is an empty String, the method should return all words in the tree. In this case the result is the same as calling `getAllWords()`.

End of examination paper