UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

1   Introduction

This document contains both Practical 6 and Assignment 6. In general the Assignments will build upon the work of the current Practical.

1.1   Submission

The Practical will be FitchFork only and is due at 22:00 on Friday the 12th of November. The Assignment will also be FitchFork only and is due at 22:00 on Friday 19th of November. You may use the practical sessions to ask for assistance if it is required.

1.2   Plagiarism policy

It is in your own interest that you, at all times, act responsible and ethically. As with any work done for the purpose of your university degree, remember that the University of Pretoria will not tolerate plagiarism. Do not copy a friend's work or allow a friend to copy yours. Doing so constitutes plagiarism, and apart from not gaining the experience intended, you may face disciplinary action as a result.

For more on the University of Pretoria's plagiarism policy, you may visit the following webpage: `http://www.library.up.ac.za/plagiarism/index.htm`

1.3 Practical component [25%]

There are two tasks you must complete for this practical. All tasks will be marked by FitchFork. All tasks in this practical and assignment will involve working with the following struct,

```
struct Trie {
        struct Trie *children[26];
        char isWord[26];
};
```

1.3.1 Task 1: Allocating and Initialising

You must implement two functions called trieAlloc and trieInit in assembly. trieAlloc must only allocate and return memory for the Trie node struct. You must allocate the correct amount of memory including padding. The trieInit function must take in a pointer of type Trie and set all pointer and isWord values in the Trie to zero.

1.3.2 Task 2: Inserting

For this task you must implement the function trieInsert in assembly. The function will accept a pointer to the root node of the trie and a word as a string. You must subsequently insert the string into the trie. As follows,

```
current = root

for each letter in word:
        let letterIndex = letter - 'a'
        if is last letter:
                current->isWord[letterIndex] = 1
        else:
                if needed create child at letterIndex
                current = current->children[letterIndex]
```

As an example if you insert the words "cat", "dog" and "door" into the trie you should have the following structure, In the image the blue, red and green lines represents "cat", "dog" and "door" respectively. Note that "dog" and "door" share 2 nodes because they both have a prefix of "do" which has length 2.

When you are finished, create an archive with your assembly files (**alloc.asm** and **insert.asm**) and upload it to the Practical 6 slot.
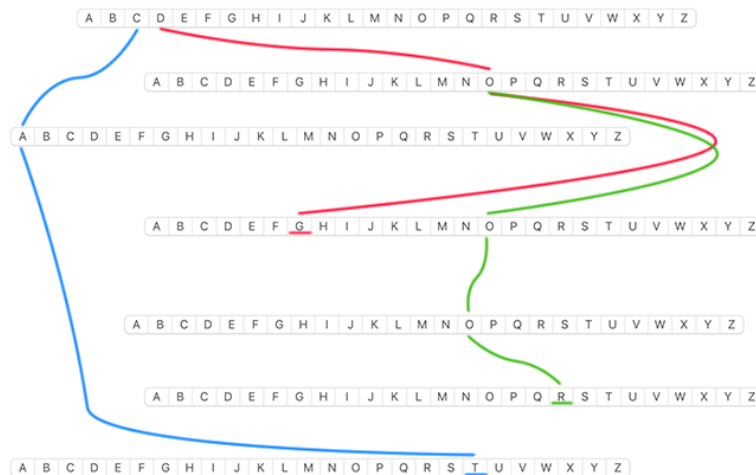
1.4   Assignment component [75%]

In the Assignment component you must implement a function to check if your trie contains a given word. Your method will be used to solve a word search problem using brute force. The word search problem solution is provided for you in the given files. It is only your responsibility to implement the trieContains function.

Your contains function will accept the root of the trie and a string for which it must check. The trie given to the function will be a trie loaded with words from the "sowpods.txt" dictionary.

Your contains function should work as follows,

```
current = root

for each letter in word:
        let letterIndex = letter - 'a'
        if is last letter:
                return current->isWord[letterIndex] == 1
        else if current->children[letterIndex] == 0:
                return 0
        else:
                current = current->children[letterIndex]
return 0
```

When you are finished, create an archive with your assembly files (**alloc.asm**, **insert.asm** and **contains.asm**) and upload it to the Assignment 6 slot.

1.5   A note on the assignment tasks:

For all assignment tasks you will be given a C wrapper class that will handle input and output for you. You must simply implement the functions that will be called from inside the class and provide the correct return values.

2 Mark Distribution

Activity	Mark
Prac Task 1	10
Prac Task 2	15
Assignment	75
Total	**100**