CSE 332: Data Structures and Parallelism

Section 1: `WorkLists`

0. Odd Jobs

For each of the following scenarios, choose the

(1) ***ADT*:** `Stack` or `Queue`

(2) **and underlying *data structure*:** `Array`, `LinkedList` with front, or `LinkedList` with front and back*

then (3) **give a reason for each decision** (think about runtime, space, and simplicity).

*i.e. front and back pointers for $\mathcal{O}(1)$ access to the front and back. Assume a singly-linked list.

(a) You're designing a tool that checks code to verify that all opening brackets, braces, parentheses have closing counterparts.

ADT:		
underlying data structure:		

(b) Disneyland has hired you to find a way to improve the processing efficiency of their long lines at attractions. There is no way to forecast how long the lines will be.

ADT:		
underlying data structure:		

(c) A sandwich shop wants to serve customers in the order that they arrived, but also wants to frequently look ahead to know what people have ordered (e.g. checking 1st person, 2nd person, ..., last person in line).

ADT:		
underlying data structure:		

1. `Trie` to Delete 0's and 1's?

(a) Insert all possible binary strings of lengths 0-3 (i.e. "", "1", "0", "10", ..., "110", "111") into a `Trie`.

(b) From here, remove all binary strings of length 2 (e.g. "00"). How many nodes would disappear? Why?

(c) From here, remove all binary strings of length 3 (e.g. "000"). How many nodes would disappear? Why?

2. Call Me Maybe

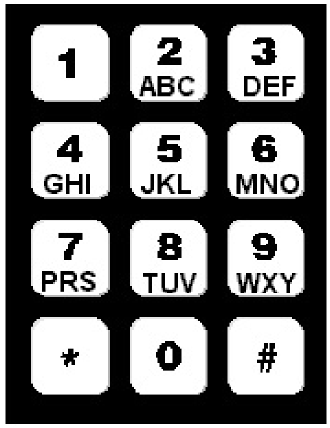
(a) Suppose you want to transfer someone's phone book to a data structure so that you can call all the phone numbers with a particular area code efficiently. What data structure would you use? How would you implement it? There are a few answers here.

(b) What is the time complexity of your solution?

(c) What is the space complexity?

3. Let’s `Trie` to be Old School

Text on nine keys (T9)’s objective is to make it easier to type text messages with 9 keys. It allows words to be entered by a single keypress for each letter in which several letters are associated with each key. It combines the groups of letters on each phone key with a fast-access dictionary of words. It looks up in the dictionary all words corresponding to the sequence of keypresses and orders them by frequency of use. So for example, the input ’2665’ could be the words `{book, cook, cool}`. Describe how you would implement a T9 dictionary for a mobile phone.



T9 Example