Computer Science CSC263H
St. George Campus

June 14, 2019
University of Toronto

Homework Assignment #3

Due: July 2, 2019, by 11:59pm

- You must submit your assignment as a PDF file of a typed (**not** handwritten) document through the MarkUs system by logging in with your CDF account at:

  `markus.teach.cs.toronto.edu/csc263-2019-05`

  To work with one or two partners, you and your partner(s) must form a group on MarkUs.

- If this assignment is submitted by a group of two or three students, the PDF file that you submit should contain for each assignment question:

  1. The name of the student who *wrote* the solution to this question, and
  2. The name(s) of the student(s) who *read* this solution to verify its clarity and correctness.

- The PDF file that you submit must be clearly legible. To this end, we encourage you to learn and use the LaTeX typesetting system, which is designed to produce high-quality documents that contain mathematical notation. You can use other typesetting systems if you prefer, but handwritten documents are not accepted.

- By virtue of submitting this assignment you (and your partners, if you have any) acknowledge that you are aware of the homework collaboration policy that is stated in the csc263 course web page: `https://github.com/csc263-summer2019/csc263-s19/wiki/Course-Information-Sheet#assignment-collaboration`.

- For any question, you may use data structures and algorithms previously described in class, or in prerequisites of this course, without describing them. You may also use any result that we covered in class, or is in the assigned sections of the official course textbook, by referring to it.

- Unless we explicitly state otherwise, you should justify your answers. Your paper will be marked based on the correctness and completeness of your answers, and the clarity, precision, and conciseness of your presentation.

Question 1. (20 marks)

a. (20 marks) Explain how you can augment an AVL tree with pointers such that it would be possible to add these 4 functions such that they run in O(1) time.

- **successor(x)** - returns node with next biggest key to x.key
- **predecessor(x)** - returns node with next smallest key to x.key
- **minimum(x)** - returns node with smallest key in subtree with root x
- **maximum(x)** returns node with biggest key in subtree with root x

In your explanation, be sure to include:

- State the extra information you are adding to augment your data structure
- Explain the how you will need to modify insertion/deletion to ensure the augmented values are updated correctly when the tree is modified ***clearly*** and ***concisely***, in English.
- Give the algorithm's pseudo code for minimum(), maximum(), successor() and predecessor().
- Explain why your algorithm achieves the required worst-case time complexity described above.

b. (0 marks) This question is purely for your own satisfaction and is worth no marks. Starting with the AVL tree code in the repository, implement your augmented AVL tree.

Question 2. (20 marks)

a. (20 marks) Suppose you were given the task of writing a program to automatically do a word search puzzle. That is you are given a list of words and a 2D array of characters. Your program must search the grid for the list of words. Words can appear in all 8 directions (up-down, down-up, left-right, right-left, and diagonally in all 4 directions). The function returns an array where each item consists of the word being searched for, the indices of the starting and ending characters of that word in the grid. If a word is found in more than one place, your result array will contain a separate array entry for each time the word was found. You may assume that the maximum length of each word is a small constant. Given a 2D array with r rows and c columns and a list of w words, describe an algorithm that can solve the problem with at most $O(rc + w)$ using a data structure we studied.

To answer this question, you must:

- State which data structure you are using, and describe the items that it contains.
- Give the algorithm's pseudo code for finding the coordinates for the words
- Explain your algorithm ***clearly*** and ***concisely***, in English.
- Explain why your algorithm achieves the required worst-case time complexity described above.

b. (0 marks) This question is purely for your own satisfaction and is worth no marks. Implement your a program that will do as you say. A pair of text files will be made in the next few days in the course repository containing some wordsearch puzzles and wordlists. They will have the following format:

```
line 1: numberOfWords row col
line 2: word 1
...
line n+1: word n
line n+2: first row of word search
...
```

Example

```
2 3 5
cat
dog
bat
abcde
fghij
klmno
```

[***The questions below will not be corrected/graded. They are given here as interesting problems that use material that you learned in class.***]

Question 3. (0 marks) Consider an abstract data type that consists of a *set* S of integers on which the following operations can be performed:

Add(i) Adds the integer i to S. If this integer already is in S, then S does not change

Sum(t) Returns the sum of all elements of S that are less than or equal to the integer t. If all the elements of S are greater than t, then return 0.

Describe how to implement this abstract data type using an augmented AVL tree. Each operation should run in $O(\log n)$ worst-case time, where $n = |S|$. Since this implementation is based on a data structure and algorithms described in class and in the AVL handout, you should focus on describing the extensions and modifications needed here.

a. Give a precise and full description of your data structure. In particular, specify what data is associated with each node, specify what the key is at each node, and specify what the auxiliary information is at each node. In particular, what is (are) the augmented field(s), and what identity should this (these) fields satisfy. Illustrate this data structure by giving an example of it (with a small set of your own choice).

b. Describe the algorithm that implements each one of the two operations above, and explain why each one takes $O(\log n)$ time in the worst-case. *Your description and explanation should be in clear and concise English.* For the operations SUM, you should also give the algorithm's high-level pseudocode.