Ann, Bob, Charlie (*replace with your names and correct the date*)
COSC 336
3/19/2500

Assignment 6

Instructions.

1. Due date and time: As indicated on Blackboard.
2. This is a team assignment. Work in teams of 3-4 students. Submit on Blackboard one assignment per team, with the names of all students making the team.
3. The exercises will not be graded, but you still need to present your best attempt to solve them. If you do not know how to solve an exercise, say it. This will give me feedback about your understanding of the theoretical concepts.
4. Your programs must be written in Java.
5. Write your programs neatly - imagine yourself grading your program and see if it is easy to read and understand.

   Comment your programs reasonably: there is no need to comment lines like "i++" but do include brief comments describing the main purpose of a specific block of lines.
6. You will submit on **Blackboard** 2 files.

   The **1-st file** is a pdf file (produced ideally with latex and Overleaf) and it will contain the following:

   (a) The solution to the Exercises (see the remark above).
   (b) A short description of your algorithm for the Programming Task.
   (c) A table with the results your program gives for the data sets indicated for the programming task.
   (d) The java code (so that the grader can make observations) of the program.

   The **2-nd file** is the .java file containing the java source code for Programming Task.

Exercise 1. Consider inserting the keys $10, 22, 31, 4, 15, 28, 17, 88, 59$ into a hash table of length $m = 11$ using open addressing with the hash function $h(x) = x \pmod{11}$ and linear probing, quadratic probing, and double probing. Illustrate the result by showing the 3 tables obtained after inserting these keys using

- linear probing,
- quadratic probing
- double hashing with $h_1(x) = x \pmod{11}$ and $h_2(x) = 7 - x \pmod{7}$.

Answer:

Exercise 2. Recall the Partition subroutine employed by QuickSort. You are told that the following array has been partitioned around some pivot element:

3	1	2	4	5	8	7	6	9

Which of the elements could have been the pivot element? (List all that apply; there could be more than one possibility.)

Exercise 3. Let α be some constant, independent of the input array length n, strictly between 0 and 1/2. What is the probability that, with a randomly chosen pivot element, the Partition function produces a split in which the size of both the resulting subproblems is at least $\alpha \cdot n$. Choose the answer from the following list and justify your answer.

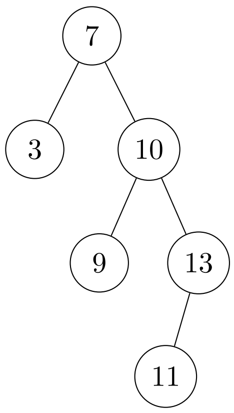
- α
- $1 - \alpha$
- $1 - 2\alpha$
- $2 - 2\alpha$

Programming task The program involves various operations for binary search trees. Your task is to modify the program at
https://www.geeksforgeeks.org/insertion-in-binary-search-tree/?ref=lbp

• Add to the class Node a data member called int size which keeps the number of nodes in the tree rooted at that node (including in the count the node itself). The constructors and the insertion function need to take into account the sizes of the nodes.

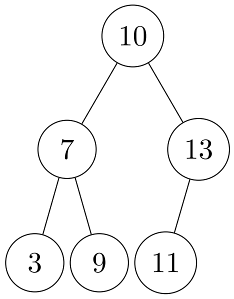
• Modify the insert function so that duplicates are also inserted (which is not done in the version at the above link). If the value x to be inserted is equal to the value in the root, insert x in the left subtree.

• Write two functions called leftRotate (Node t), which rotates the root t to the left, so that the right child of t (if there is one; otherwise the rotation does not do anything) becomes the parent of t, and symmetrically rightRotate (Node t) which rotates the root to the right. See Figure 13.2, page 313 in the textbook, or Notes 6 on Blackboard. Note that when you do leftRotate (Node t), you need to change the size of t and of t.right, and similarly for rightRotate.



For instance for the tree in the figure, node 7 has size 6, node 3 has size 1, node 9 has size 1, node 10 has size 4, node 13 has size 2, and node 11 has size 1.

If we do leftRotate for the root we get the tree



To test the augmented class, you will insert some values and then do a *preorder* traversals and print in this order the pairs (value, size) of all the nodes. Next, you leftRotate the root and print again the tree after rotation. In other words, you do a preorder traversal **before and after** the rotation, but print the elements in this order.

Test data 1: insert 7, 10, 3, 9, 13, 11. Your program will print: (7,6), (3,1), (10,4), (9,1), (13, 2), (11,1). Next, do a leftRotate, and print the tree after rotation and you get

(10,6), (7,3), (3,1), (9,1) (13,2), (11, 1).

Test data 2: Insert one by one the numbers in the file input-6.1. The first line contains how many numbers are there (there are 1000 numbers), and the next line contains the list of numbers.

Report in the pdf file the first 25 pairs (you do complete preorder traversals before and after but report only the first 25 elements).

Test data 3: Insert one by one the numbers in the file input-6.2. The first line contains how many numbers are there (there are 10000 numbers), and the next line contains the list of numbers.

Report in the pdf file the first 25 pairs (as above).