Concordia University
Department of Computer Science and Software Engineering
Comp 352
Data Structure and Algorithms
Assignment 3 – Summer 2021
Due date: Friday June 18th, 2021 by midnight

Heads-Up

- The programming questions can be done individually or in a team of two members max. However, the written questions must be completed and submitted individually.
- For the written questions, you must submit the answers to all the questions. However, only a couple of questions chosen at random, will be corrected and will be evaluated to the full 50 marks.
- For the programming questions you are not allowed to use any java build-in classes/methods/algorithm unless indicated.

1. Written Questions (50 marks)

Question 1

Assume a hash table utilizes an array of 13 elements and that collisions are handled by *separate chaining*. Considering the hash function is defined as: h(k) = k mod (13).

a) Draw the contents of the table after inserting elements with the following keys:
{245, 28, 10, 49, 70, 225, 122, 12, 180, 140, 177, 65, 223, 85, 111, 256, 18, 69, 59, 185, 105, 120, 44}.

b) What is the total number of collisions caused by the above insertions?

Question 2

Assume an *open addressing* hash table implementation, where the size of the array $N = 19$, and the *double hashing* is performed for collision handling. The second hash function is defined as:
$d(k) = q - k \bmod q$, where k is the key being inserted in the table and the prime number $q = 11$. Use simple modular operation ($k \bmod N$) for the first hash function.

a) Show the content of the table after performing the following operations, in order:
put(37), put(17), put(24), put(36), put(62), put(28), put(58),put(47), put(19).
b) What is the size of the longest cluster caused by the above insertions?
c) What is the number of occurred collisions as a result of the above operations?
d) What is the current value of the table's *load factor*?

Question 3

Assume the utilization of *linear probing* instead of *double hashing* for the implementation given in Question 2. Still, the size of the array $N = 19$, and that simple modular operation (k mod N) is used for the hash function.

a) Show the contents of the table after performing the following operations, in order:
put(37), put(17), put(24), put(36), remove(62), remove(28), put(58), put(47), remove(19).

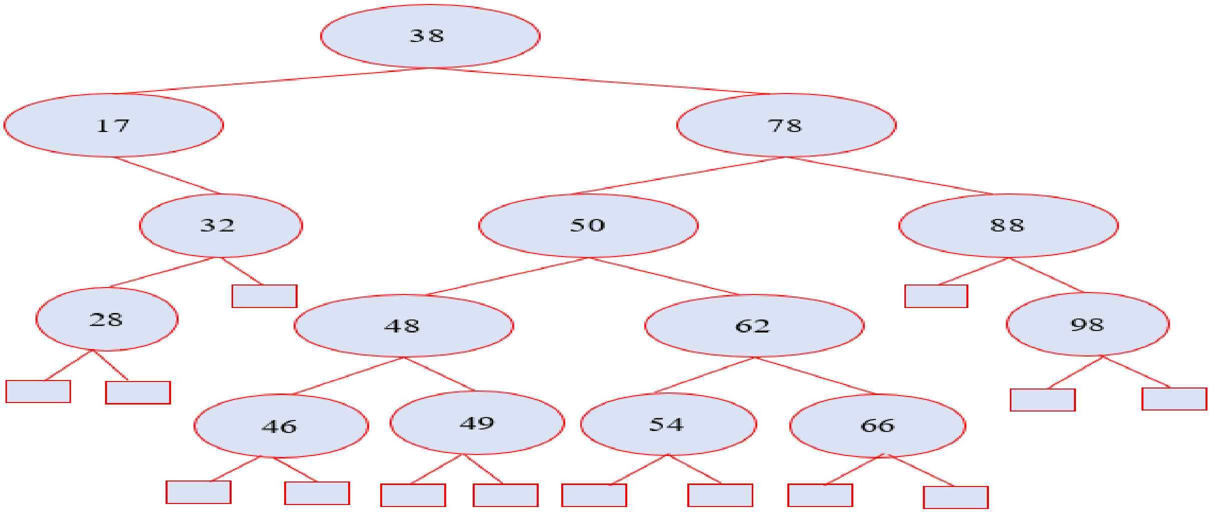
b) What is the size of the longest cluster caused by the above insertions? Using Big-O notation, indicate the complexity of the above operations.

c) What is the number of occurred collisions as a result of the above operations?

Question 4

Note: Part (a) and (b) are independent in this question.

Consider the following AVL tree:



a) Draw the AVL tree resulting from the insertion of an entry with key 56 in the AVL tree shown above.

b) Draw the AVL tree resulting from the removal of the entry with key 28 in the AVL tree shown above.

Question 5

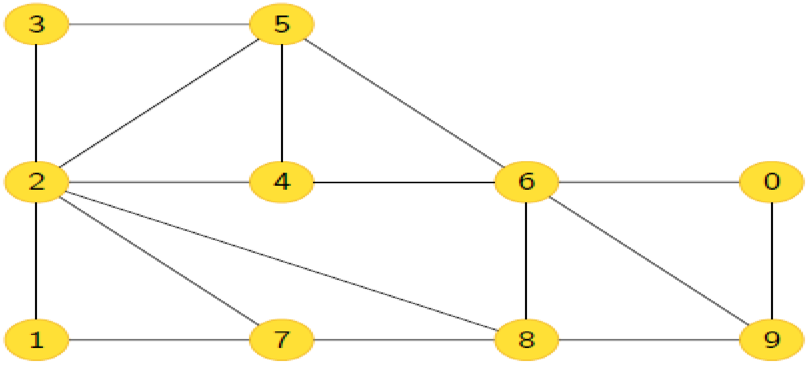
Consider the following elements:
12, 47, 74,19, 89, 4, 63, 26, 53, 8, 93, 71, 15, 87, 50, 17, 82

Trace the steps when sorting these values into ascending order using:

a) Merge Sort
b) Quick Sort (using (middle +1) element as pivot point)
c) Bucket Sort – We know that the numbers are less than 99 and there are 10 buckets.
d) Radix Sort

Question 6

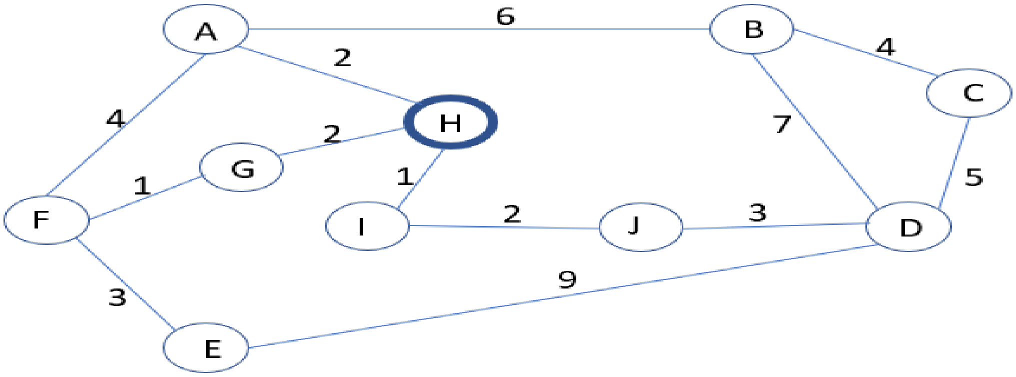
Consider the graph shown below:



1. Give the adjacency matrix representing this graph.
2. Give the adjacency lists representing this graph.
3. Show the breadth-first search trees for the graph starting at node 0.
4. Show the depth-first search trees for the graph starting at node 0.

Question 7

Use *Dijkstra's* Algorithm to find the shortest path of the following graph from node H to each of the other nodes.



Programming Questions (50 marks)

In these programming questions you will experimenting writing your own Binary Search Tree with a couple of methods.

A *Binary Search Tree* (BST) is a tree-based data structure in which each node has at most two children, which are referred to as the left child and the right child, and the topmost node in the tree is called the root. It additionally satisfies the binary search property, which states that the key in each node must be greater than or equal to any key stored in the left subtree, and less than or equal to any key stored in the right subtree.

Important Note: For all the questions below you must provide a solution that does not refer to the given example. (i.e. it should be able to output the appropriate output for any given collection of integer keys).

Question 1:

In this first question, you will be developing your own Binary Search Tree *MyBST* program (i.e., class) in two versions.

a) In version 1 of your *MyBST*, you implement a recursive method to insert random integer keys in the BST.

b) In version 2 of your *MyBST*, you implement an iterative method to insert random integer keys in the BST.

Your program should take a set of random integer keys such as:

15 25 20 22 30 18 10 8 9 12 6
and output the corresponding BST:
6 8 9 10 12 15 18 20 22 25 30

Question 2:

In this question you will be using one version of your *MyBST* program from question 1, and add a method that finds the total subtrees of the BST that are within the given range.

Example, given a set of random integer keys such as:
18 23 12 28 15 9 33 25 13 11 21
and the range [8, 23]
Your program should output: The total number of subtrees is 6.

Figure 1 depicts the details of this results:

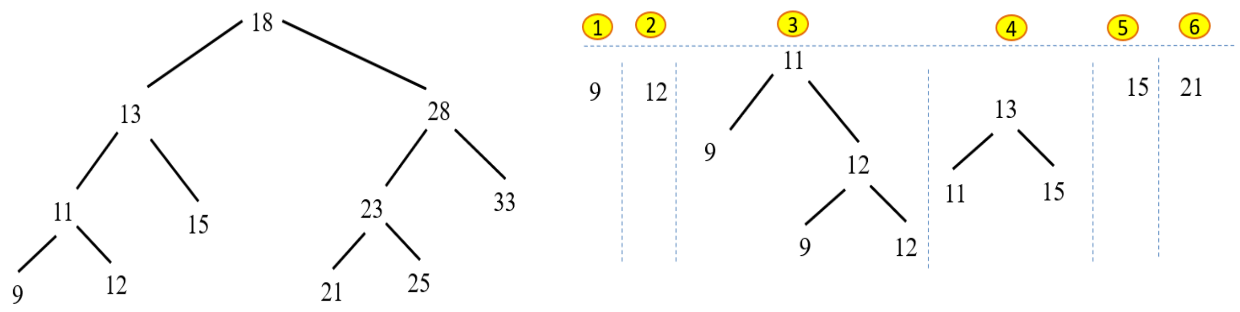


Figure 1

Question 3:

In this question you will be using one version of your *MyBST* program from question 1, and add a method that remove nodes from your BST that have keys outside a given valid range.

Example, given a set of random integer keys such as:
22, 48, 19, 4, 13, 78, 83, 59, 29, 17, 66 and the range [5,40]

Your program should output: The key within the valid range are:
13 17 19 22 29.

Important Requirements :

1. Do not use any java abstract data type or packages when answering these programming questions of your assignment.
2. Your code should handle boundary cases and error conditions. It is also imperative that you test your classes/methods.
3. For these programming questions, you are required to submit the commented Java source files, the compiled files (.class files), and the test run text files.

Guidelines

For all questions, including the programming questions, the parts that require written answers, pseudo-code, analysis, table, etc., you can express your answers into any number of files of any format, including image files, plain text, hand-writing, Word, Excel, PowerPoint, etc. However, no matter what program you are using, you must convert each and every file into a PDF before submitting. This is to ensure the markers and you have exact same view of your work regardless of the original file formats.

1. Written Questions

You must submit a copy of your written answers as A#studentID under A#3 WrittenQuestions submission on Moodle, (studentID denotes your student ID, and A# is A3 for this third assignment).

2. Programming Questions

Developing your programming work using a Java IDE such as NetBeans, Eclipse, etc., you are required to submit all the source code files created by your Java IDE, together with any input files used and output files produced by your program(s).

You must submit a copy of your programming questions as a zip file that includes the sources java files, input/output files, written pseudo code, etc., under A#3 ProgrammingQuestions submission on Moodle,

a) If you are working individually, name your zip file A#studentID, where studentID denotes your student ID, and A# is A3 for this first assignment.
b) If you are in a team, you must submit only ONE copy of your programming part, name your zip file A#studentID1_studentID2, where studentID1 denotes the student ID of the member responsible for submitting the programming solutions for both students.

Last but not Least

To receive credit for your programming solutions, you must demo your program to your marker, who will schedule the date and time for the demo for you (please refer to the course outline for full details). If working in a team, both members of the team must be present during the demo. Please notice that failing to demo your programming solution will result in zero mark regardless of your submission.