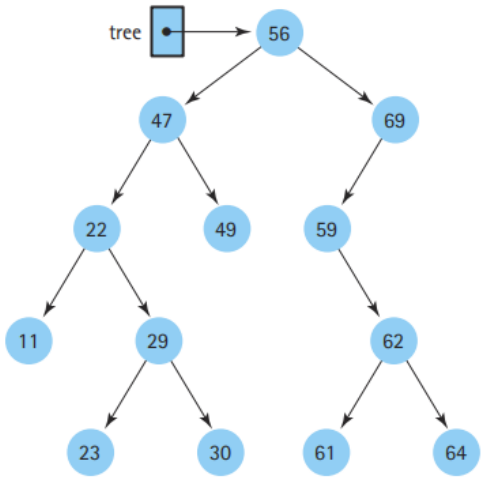
Homework 3 CSE 225

1. Draw the binary search tree whose elements are inserted in the following order:

   50 72 96 94 107 26 12 11 9 2 10 25 51 16 17 95

2. Answer the following questions using the tree below-



i)
a. What is the height of the tree?
b. What nodes are on Level 3?

ii) Show the order in which the nodes in the tree are processed by
a. an inorder traversal of the tree.
b. a postorder traversal of the tree.
c. a preorder traversal of the tree.

iii) Show how the tree would look after the deletion of 29, 59, and 47.

3. a. Extend the Binary Search Tree ADT to include the member function **LeafCount** that returns the number of leaf nodes in the tree.

   b. Extend the Binary Search Tree ADT to include the member function **SingleParentCount** that returns the number of nodes in the tree that have only one child.

4. For the questions below, use the following values and draw the hash tables:
66 47 87 90 126 140 145 153 177 285 393 395 467 566 620 735

i. Store the values into a hash table with 20 positions, using the division method of hashing (mod operator) and the linear probing method of resolving collisions.

ii. Store the values into a hash table with 20 positions, using rehashing as the method of collision resolution. Use **key % tableSize** as the hash function, and (**key + 3) % tableSize** as the rehash function.

iii. Store the values into a hash table with ten buckets, each containing three slots. If a bucket is full, use the next (sequential) bucket that contains a free slot.

iv. Store the values into a hash table that uses the hash function **key % 10** to determine into which of ten chains to put the value

5. A priority queue containing characters is implemented as a heap stored in an array. The precondition states that this priority queue cannot contain duplicate elements. Currently, the priority queue holds 10 elements as shown. What values might be stored in array positions 7–9 so that the properties of a heap are satisfied?

pq.items.elements

[0]	Z
[1]	F
[2]	J
[3]	E
[4]	B
[5]	G
[6]	H
[7]	?
[8]	?
[9]	?

6. Use the following specification of an *undirected graph*.
EmployeeGraph = (V, E)
V(EmployeeGraph) = {Susan, Darlene, Mike, Fred, John, Sander, Lance,
Jean, Brent, Fran}
E(EmployeeGraph) = {(Susan, Darlene), (Fred, Brent), (Sander, Susan),
(Lance, Fran), (Sander, Fran), (Fran, John), (Lance,
Jean), (Jean, Susan), (Mike, Darlene), (Brent, Lance),
(Susan, John)}

i. Draw a picture of **EmployeeGraph**.

ii. Draw **EmployeeGraph**, implemented as an adjacency matrix. Store the vertex values in alphabetical order.

iii. Using the adjacency matrix for **EmployeeGraph** , describe the path from Susan to Lance
   a) using a breadth-first strategy.
   b) using a depth-first strategy.

iv. Which one of the following phrases best describes the relationship represented by the edges between the vertices in **EmployeeGraph**?
   a) "works for"
   b) "is the supervisor of"
   c) "is senior to"
   d) "works with"

7. Extend the class **GraphType** in this chapter to include a Boolean **EdgeExists** operation, which determines whether two vertices are connected by an edge.
a. Write the declaration of this function. Include adequate comments.
b. Using the adjacency matrix implementation developed in the chapter and the declaration from part (a), implement the body of the function.

8. True or false? Correct any false statements.
a. A binary search of a sorted set of elements in an array is always faster than a sequential search of the elements.
b. A binary search is an O(*N* log2*N*) algorithm.
c. A binary search of elements in an array requires that the elements be sorted from smallest to largest.
d. When a hash function is used to determine the placement of elements in an array, the order in which the elements are added does not affect the resulting array.
e. When hashing is used, increasing the size of the array always reduces the number of collisions.
f. If we use buckets in a hashing scheme, we do not have to worry about collision resolution.
g. If we use chaining in a hashing scheme, we do not have to worry about collision resolution.
h. The goal of a successful hashing scheme is an O(1) search.