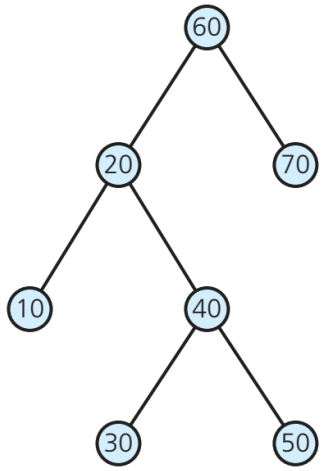
Binary search trees

A – Theory part

A.1. Consider the binary search tree shown aside and answer the following questions.



a. What node or nodes are
   - The tree's root?
   - Pairs of parents - children?
   - Siblings?
   - Ancestors of 50?
   - Descendants of 20?
   - Leaves?

b. What is the height of the tree?

c. Starting with an empty binary search tree, in what order should you insert items to get the binary search tree shown above?

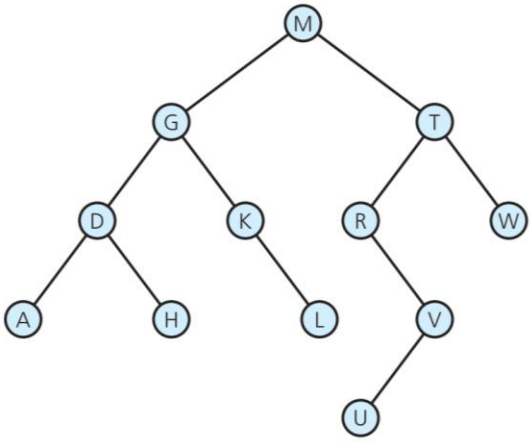
d. Trace the search algorithm when it searches for 30. Similarly, for 15. In each case, list the nodes in the order in which the search visits them.

e. What tree results after you insert the entries 80, 65, 75, 45, 35, and 25, in that order?

f. After inserting the nodes mentioned in part e, what tree results when you remove the entries 50 and 20.

A.2. Consider the binary search tree shown aside and answer the following questions.

A.3. Consider the binary tree shown aside and answer the following questions.



a. Is this a binary search tree? Explain.

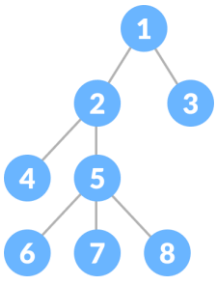
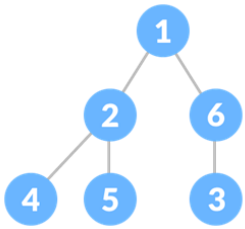
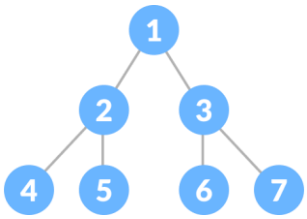
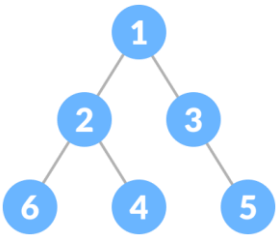
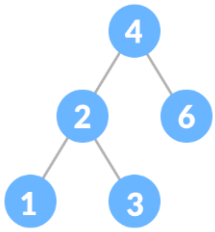
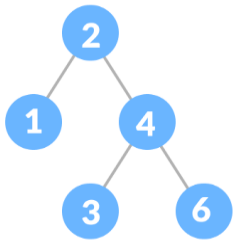
b. Perform in-order, pre-order and post-order

c. If it is not identified as a binary search tree, re-arrange some branches to make it a binary search tree. Then, what does the tree look like after you remove M, D, G, and T, in that order?

A.4. Beginning with an empty binary search tree, what binary search tree is formed when you insert the following values in the order given?

a. W, T, N, J, E, B, A b. W, T, N, A, B, E, J c. A, B, W, J, N, T, E d. B, T, E, A, N, W, J

A.5. Which of the following are binary trees? If a tree is a binary tree, check whether it is a binary search tree? A full binary tree? A complete binary tree?

a.		b.		c.	
d.		e.		f.	

A.6. Consider a binary search tree that contains integers from 1 to 100. Which of the following is possibly a sequence of keys checked during the search of key **45**? Explain why the other choices are impossible.

- ☐ 5, 2, 1, 10, 39, 34, 77, 63
- ☐ 1, 2, 3, 4, 5, 6, 7, 8
- ☐ 50, 25, 26, 27, 40, 44, 42
- ☐ 50, 25, 26, 27, 40, 44

A.7. Consider a binary search tree that contains integers from 1 to 1000. Which of the following is NOT a sequence of keys checked during the search of key **363**? Explain why the other choices are rejected.

- ☐ 2, 252, 401, 398, 330, 344, 397, 363
- ☐ 925, 202, 911, 240, 912, 245, 363
- ☐ 924, 220, 911, 244, 898, 258, 362, 363
- ☐ 935, 278, 347, 621, 299, 392, 358, 363
- ☐ 2, 399, 387, 219, 266, 382, 381, 278, 363

A.8. Differentiate full binary tree and complete binary tree. In some other materials, there is also a perfect binary tree. Differentiate perfect binary tree from the two previous binary tree.

A.9. How many differently shaped n-node binary trees are possible? How many differently shaped n-node binary search trees are possible? (Write recursive definitions.)

A.10. By using mathematical induction, prove that

a. A full binary tree of height $h \geq 0$ has $2^h - 1$ nodes.

b. The maximum number of nodes in a binary tree of height h is $2^h - 1$.

c. A binary tree with n nodes has exactly n + 1 empty subtrees.

B – Coding part

B.1. Consider a binary search tree whose every node contains an integer key. No duplicate value is allowed. Implement an array-based binary search tree with the following operations.

- Define a data structure to represent a node in the binary search tree
- Test whether a binary search tree is empty
- Get the height of a binary search tree
- Get the number of nodes in a binary search tree
- Get the data in a binary search tree's root
- Insert a new item into a binary search tree
- Remove the given item from a binary search tree
- Remove all entries from a binary search tree
- Retrieve the given item from a binary search tree
- Test whether a binary search tree contains a specific entry
- Traverse the items in a binary search tree in pre-order, in-order, or post-order

B.2. Repeat the above question with a link-based implementation of the binary search tree.

B.3. Write a program that maintains a database containing data, such as name and birthday, about your friends and relatives. You should be able to enter, remove, modify, or search this data. Initially, you can assume that the names are unique. The program should be able to save the data in a file for use later.

Design a class to represent the database and another class to represent the people. Use a binary search tree of people as a data member of the database class. You can enhance this problem by adding an operation that lists everyone who satisfies a given criterion. For example, you could list people born in a given month. You should also be able to list everyone in the database.