Department of Computer Science
Faculty of Computing and Information Technology
King Abdulaziz University, Jeddah, Saudi Arabia.

Assignment 3: Binary Search Tree

Assigned on Thursday October 27, 2022
[Deadline: Thursday November 10, 2022]

Objectives

1. Apply search, insert, and delete operations on trees.
2. Comprehend tree concept and tree operations.
3. Implement the concept of the tree and tree operations.

ABET Student Outcome (SO-2):
Design, implement, and evaluate a computing-based solution to meet a given set of computing requirements in the context of the program's discipline.

Read Carefully:
This assignment is worth **5%** of your final grade.

NOTE:
This is an individual assignment; you must solve it by yourself. Any form of cheating will result in receiving zero in the assignment.

The deadline for this assignment is **Thursday November 10, 2022 @ 11:59pm**. Once the clock becomes 11:59 PM, the submission will be closed!

LATE SUBMISSION: No assignment will be accepted after the deadline
Blackboard Submission:

This assignment must be submitted online via Blackboard.
The source file(s) of your program should be zipped up. You must name the zip file using the following naming convention: **SectionNumber_StudentID_ProgramNumber.zip**
Example: EA_1110348_P3.zip

Question:	1	2	3	Total
Points:	20	20	60	100

Concept Application & Algorithmic Part

Question 1: (20 points)

Algorithm Write up. Write an algorithm that returns *true* if the total number of nodes in the left and right subtrees in a given binary tree are equal. The algorithm should return *false* otherwise (i.e., the total number of nodes in the left and right subtrees are not equal).
For example, the algorithm for tree in figure (1) will return false, whereas for tree in figure (2), the algorithm will return true.

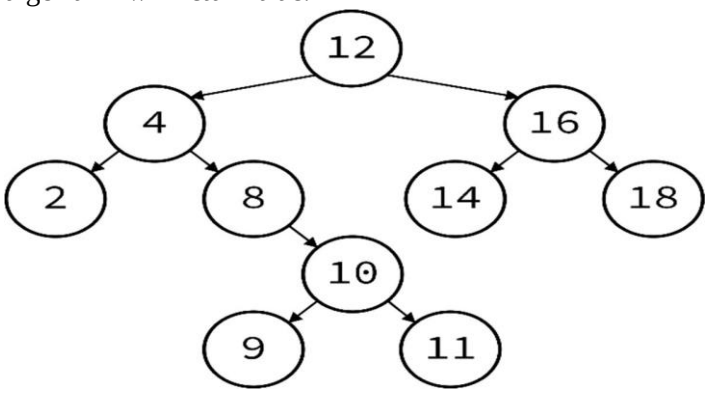


Figure (1)

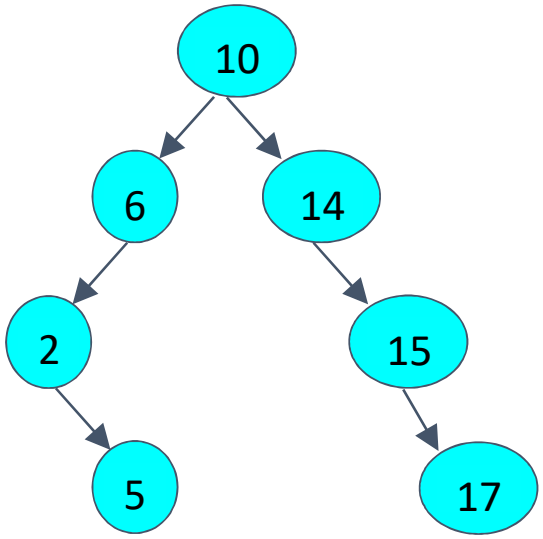
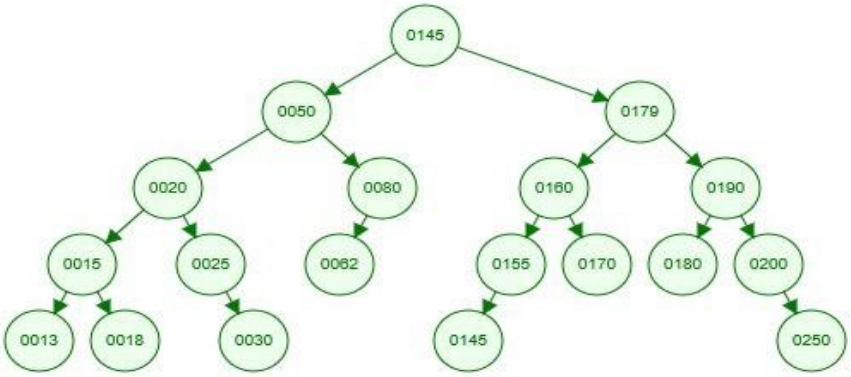


Figure (2)

Input:
Output:
Algorithm:

Question 2: (20 points)

Given the following Binary Search Tree, show the complete tracing of **recurSearch**() method (given in the code of BST) to find the node 0160. You are required to draw recursive trees or boxes to show the detailed working of this method. Show the output and the detailed working of this method.



Question 3:(60 points)

Objective

The primary objective of this program is to learn to implement binary search trees and to combine their functionalities with linked lists.

Program Description

Uber System is a simulation software to simulate a variety of Uber oriented functionalities for captains. The system is capable of adding records for captains, booking rides, finishing a ride, displaying information and rating captains.

You are required to write a program with the following methods:

1. a function to add a new captain by adding a new node to the binary search tree.
2. a function to search for a specific captain by his/her ID.
3. a function that prints all captains' names with their specified IDs and rating stars.
4. a function that books a ride.
5. a function that finishes a ride. The method calculates the captain's stars based on the rider satisfaction.
6. a function that removes a particular captain by removing their nodes from the tree.

The program requires two files: one input file and one output file. The description of these files is as follow:

- The commands for the system are found in the input file which is called ***input.txt.*** The commands in this file are as follow:
  - **ADD_CAPTAIN:** This command generates a BST node for each captain in the tree. The command followed by two values: the captain ID and the captain's name. The default value for the *rating stars* will be "0"and the default value of *available* is "true".
  - **DISPLAY_CAPTAIN_INFO:** This command has one value which is the captain ID. It will output the name and the rating stars for the specified captain. If the specified captain is not found, it will output the "**Couldn't find any captain with ID number <ID >**" message.
  - **DISPLAY_ALL_CAPTAINS:** This command will output all the captains with the captains' IDs, names, available and rating stars. (Similar to the output in the ***output.txt*** file).
  - **BOOK_RIDE:** This command has one value which is the captain ID. The command will book a ride with the specified captain by changing the *available* to "false". If the *available* for the specified captain is already "false", then output the "**The captain <name> is not available. He is on another ride!"** message. If the specified captain is not found, it will output the **"Couldn't find any captain with ID number <ID >**" message.
  - **FINISH_RIDE:** This command has two value which is the captain ID, and the rider satisfaction (0 or 1). This command will make the specified captain *available* again by changing the *available* to "true". The rating stars will affect the captain rating stars. The rating stars will increase by one if the rider is satisfied and decrease by one otherwise. (Note: the rating stars is a value

between 0 and 5 only). If the *available* for the specified captain is already "true", then output the "**The captain <name> is not in a ride!"** message. If the specified captain is not found, it will output the "**Couldn't find any captain with ID number <ID >**" message.

- **DELETE_CAPTAIN:** This command has one value that represents the captain's ID. The command will search for this captain in the tree then remove his/her node from the BST, and it will output the "**The captain <name> left Uber"** message. If the specified captain is not found, it will output the "**Couldn't find any captain with ID number <ID >**" message.
- **QUIT:** This command will stop the program.

- The output of the program should be written to the file ***output.txt.*** Please use the provided output file as a reference.

Implementation

For this program, you will create the following classes:

- *CaptainNode.java*: This class will be used to create objects of type captain and it will store each captain's information.
- *UberTree.java*: All the methods will be implemented in this class.
- *MainProgram.java*: This is the class that contains the main.

Sample Input & Output File

We have provided you a sample for one input file and one output file.

*****WARNING*****

Your program MUST adhere to the EXACT format shown in the sample output file (spacing capitalization, use of dollar signs, periods, punctuation, etc). The graders will use large input files, resulting in large output files. As such, the graders will use text comparison programs to compare your output to the correct output. If, for example, you have two spaces between in the output when there should be only one space, this will show up as an error even though you may have the program correct. You will get points off if this is the case, which is why this is being explained in detail. The minimum deduction will be 10% of the grade, as the graders will be forced to go to the text editing of your program in order to give you an accurate grade. Again, your output MUST ADHERE EXACTLY to the sample output.

Grading Details

Your program will be graded upon the following criteria:

1) Adhering to the implementation specifications listed on this write-up.
2) Your algorithmic design.
3) Correctness.
4) **Use of the three classes, as specified. If your program is missing these elements, you will lose marks.**
5) The frequency and utility of the comments in the code, as well as the use of white space for easy readability. (If your code is poorly commented and spaced and works perfectly, you could earn as low as 80-85% on it.)

6) Compatibility to the **newest version** of NetBeans. (If your program does not compile in NetBeans, you will get a large deduction from your grade.)
7) Your program should include a header comment with the following information: your name, **email**, account number, section number, assignment title, and date.
8) Your output **MUST adhere to the EXACT** output format shown in the sample output file.