CS 284: Endterm

Student Name:

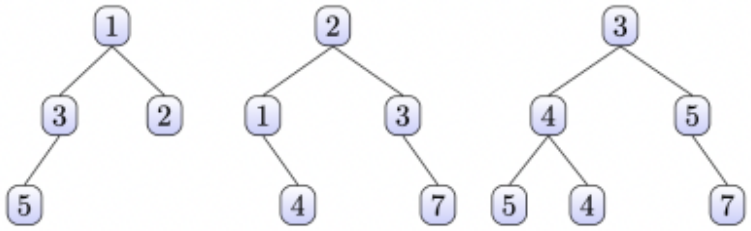
Honor Pledge:

Grade sheet:

Problem 1 (33 points)	
Problem 2 (34 points)	
Problem 3 (33 points)	

Problems

Problem 1. (Trees, 33 points) Implement a method public `BTree<Integer> sumTree(BTree<Integer> t2)` that returns a new binary tree resulting from adding the tree recipient of the message (i.e. the one referred to by `this`) and t2. For example, the sum of the first two trees below is the third one:



In this problem, you need to implement the function `sumTree` in the file `BTree.java`.

Problem 2. (Linked list, 34 points) Susan needs to build a 2-dimensional salt water pool. To build the pool, she collects the following materials: (1) a series of stone bars of different heights; (2) a series of salt bars of different heights. She puts the stone bars and salt bars in a row, which is represented by a linked list:

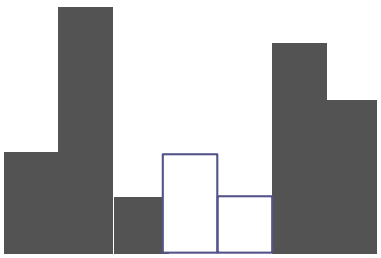
- `LinkedListofBars;`

Each node of the linked list is a node `Bar`, which contains the type (i.e., the data field `type`) and the height (i.e., the data field `height`) of the bar. The value of `type` is either '`stone`' or '`salt`', and the height is always a positive integer, you can assume every stone bar has a unique height.

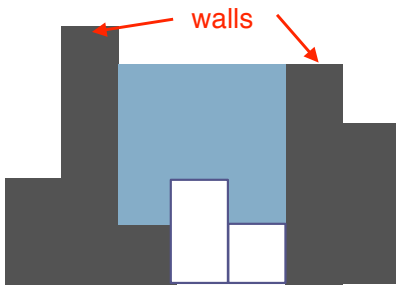
```
private static class Bar {
    private String type;
    private Integer height;
    private Bar next;

    private Bar(String bar_type, Integer bar_height) {
      type = bar_type;
      height = bar_height;
    }
}
```

Example 1. An example of the linked list of bars look like the figure below, which is `(stone, 2)`→ `(stone, 5)` → `(stone, 1)` → `(salt, 2)` → `(salt, 1)` → `(stone, 4)` → `(stone, 3)`.

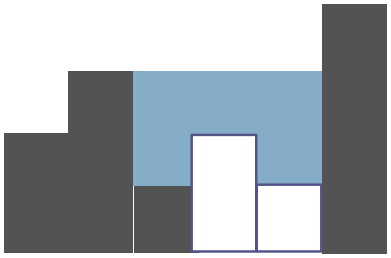


Susan uses the *tallest* and the *second tallest stone bars* in the linked list to form the two walls of the tank, and store water in between. For example, in the above pool, she fills in water between bar `(stone, 5)` and `(stone, 4)`, so the pool becomes:



How the mass of water and concentration are calculated. For a stone bar, the water mass is equal to the water level subtract the stone bar's height; for a salt bar, the salt will resolve in water, so the water mass is equal to the water level. In the above example, the water mass = (4 - 1) + 4 + 4 = 11, whereas the salt mass = 1 + 2 = 3; meanwhile, the concentration is calculated as the salt mass / the water mass, so the concentration = 3 / 11 = 0.2727.

Example 2. The second example of the linked list of bars look like the figure below, which is `(stone, 2)`→ `(stone, 3)` → `(stone, 1)` → `(salt, 2)` → `(salt, 1)` → `(stone, 4)`. Similarly, the water mass = (3 - 1) + 3 + 3 = 8, and the salt mass = 1 + 2 = 3, thus the concentration = 3/ 8 = 0.375.



Now Susan needs your help to calculate the mass of water in the pool as well as the concentration of salt in the following 3-step process:

1. `findTallest`: Finding the tallest bar by setting the class variables `tallestNode`, `tallestIdx`, and `topHeight`;
2. `findSecondTallest`: Finding the 2nd tallest bar by setting the class variables `secondTallestNode`, `secondTallestIdx`, and `secondHeight`;
3. `computeMassAndConcentration`: Computing the mass and concentration. Return a double array of size 2, the first element being the water mass, the second element being the concentration.

Problem 3. (Heaps, 33 points) Given two equally sized string arrays `(A, B)`, both of size `N`. All big integers in the arrays `A` and `B` are of the same length. Each big integer is stored in the form of String.

A concatenated number is made by concatenating any big integer from array `A` with any big integer from array `B` (`A` followed by `B`). Among all the `N^2` big integers, find out the `K` largest ones that can be divided by 3.

Example 1. `N=3, A=[1,2,3], B=[4,5,6], K=2`. The answer is [36, 24].

Example 2. `N=3, A=[4,5,6], B=[1,2,3], K=2`. The answer is [63, 51].

You need to implement the 3 functions below in the file `ConcatNumbers.java`:

- `compare(String numberA, String numberB)` in the class `The_Comparator`: Compare two large integers `numberA` and `numberB`. Return -1 if `numberA >numberB`, return 1 if `numberA <numberB`, return 0 if they are equivalent. You are not allowed to convert them to numbers.
- `check_concatenation_dividable_by_three(String numberA, String numberB)`: Check whether the large integer concatenated by numberA and numberB is divisible by 3.
- `KMaxConcatenations(String[] A, String[] B, int N, int K)`: Return a string list which contains the maximum `K` concatenated numbers which can be divided by 3. In this function, you can use a max heap to store the big integers and return the `K` largest ones. You can create a max heap by using the Java `PriorityQueue` like the following code, then add the concatenated numbers to the function:

```
PriorityQueue<String> pq = new PriorityQueue<String>(new The_Comparator());
```