Notice

- Most people infected with COVID-19 feel lethargic for one week, during which they cannot work. If you leave the assignment to the last week but fall victim to the misfortune of catching COVID-19, it is very unlikely you can finish it. You are thus advised to plan your work early.

  Please be reminded: We cannot accept late submissions for **any excuse**.

- For each question, write the following information as a comment to the method:

  1) all the Internet resources you've used for solving the problem (even though you didn't explicitly copy from them);
  2) all the students with whom you have discussed this question; and
  3) running time of your method, with the smallest possible function.[1]

- Make your algorithms as efficient as possible, in terms of both time and space, with priority on time. Your scores depend on their efficiency.

- No marks shall be awarded if your submission does not compile.[2]

- Submission procedure (each deviation will result in deduction of 10 points):

  1) Name the .java file as `<class_name>_<your_id>_<your_name>.java`, e.g., `PolyuTree_12345678d_ChanYickShun.java` if your name is (Eason) Chan Yick Shun and your ID is 12345678d. (You may find Alt-Shift-R helpful if you're using Eclipse.)
  2) **Submit the java file directly.** This is different from the previous assignment.

[1] A frequent question is "Whether I can use methods from Java library?" Most library methods are complicated and difficult to analyze. If you use them, you have to count their steps, which is almost impossible from my experience.

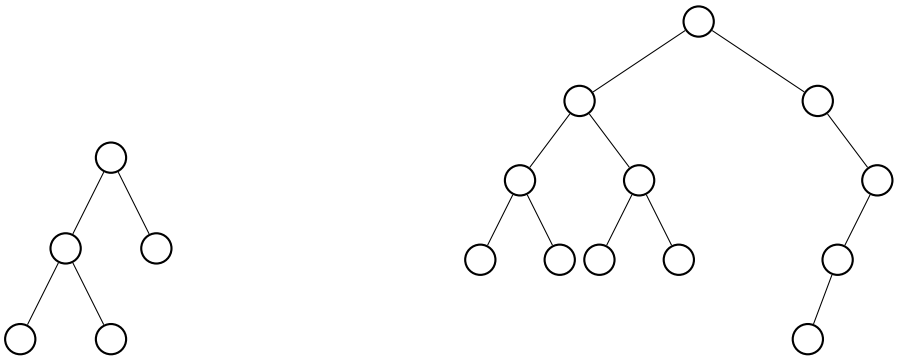
[2] If you've problems, this and this web pages may help, and you're welcome to seek help on the discussion forum (DON'T SHARE YOUR CODE).

1. (100 points) Implement a binary search tree to store students of the University. The keys are the students' names, and a surname is listed before a given name. Your implementation should allow the user to find information about a student quickly.

   The first part is about the shape of the tree you have built. Although we do not have a definitive measure for how close a binary tree is to a perfect one, the following three numbers give some partial information.

   - The largest difference between the depths of the two subtrees of a node.
   - The largest difference between the sizes of the two subtrees of a node.
   - The total number of nodes that have only one child.

   For example, they are 1, 2, and 0 for tree (a) and 3, 3, and 3 for tree (b).



(a) (b)

   The second part is about search. It supports three ways of search.

   - Search for a student with a specified name. It must be an exact match of the full name. If there are multiple students of this name, you may return any of them.
   - Search for all the students with the specified surname. For simplicity, let us treat the first word as the surname. (Sorry if your surname is "Au Yeung.")
   - Search all the students who are taking a certain class.

   Implement the following seven methods. Do not modify their signatures. You may add as many additional private helper methods as needed.

```
public void insert(Student s)
public int maxDepthDiff()
public int maxSizeDiff()
public int nodesWithOneChild()
public Student searchFullname(String name)
public Student[] searchSurname(String surname)
public Student[] searchClass(String[] roster)
```

   Write the running time as a function of d (the depth of the tree), n (the total number of students), and c (the number of students in a class). Generally, $c < \sqrt{n}$.

   You get 10 bonus points if your implementation is self-balancing.