ESO207 Theoretical Assignment 2 Semester II, 2016-17

Due Date: 9th March, 2017 **Maximum Marks: 50**

Instructions

- For questions where you need to design an algorithm, give the pseudocode, proof of correctness and time complexity of the algorithm.
- For this assignment you need to work in groups. The group will the same as earlier. Both members of the group must upload a copy of the assignment.
- You must mention in your assignment the name of your group partner as well.
- You need typeset your assignment on a typesetting system such as LaTeX(recommended) or Word.
- Upload a soft copy of the assignment on moodle by the due date.

---

1. (10 points) You are given an infinite array in which the first n cells contain a sequence of integers sorted in increasing order and the remaining cells are empty. You are **not** given the value of n. Now given an integer s design an $O(\log n)$ time algorithm that outputs whether s is present in the array or not.

2. (15 points) A proprietor owns a series of buildings next to one other, in a newly developed part of the city, and wants to put an advertisement of sale across his buildings on front side (assume all buildings are front-facing in the same direction), such that visibility is maximized. So, he is interested in knowing the banner with the largest area (rectangular banner) he can put across the buildings. However, there are certain constraints on how a banner can be put across buildings:

   - All buildings are rectangular in shape and contiguous to each other
   - No part of banner must cross top or bottom of any building (or sides for buildings on both ends)

   Note that buildings have varying widths and heights. (Ignore the mandatory gap in-between buildings, for sake of simplicity) Can you help the proprietor to find the area of largest banner he can hang across his buildings?
   Given the height and width of the buildings as input, what is the best algorithm (minimum time complexity) you can come up with to compute the largest size of such a banner? Give the time complexity in $O()$ notation.

3. (12 points) Attempt the following questions and give all the steps:

   - Draw neatly the red-black tree after deleting key 55 from it.
   - Draw the tree after inserting 34 in the original tree.

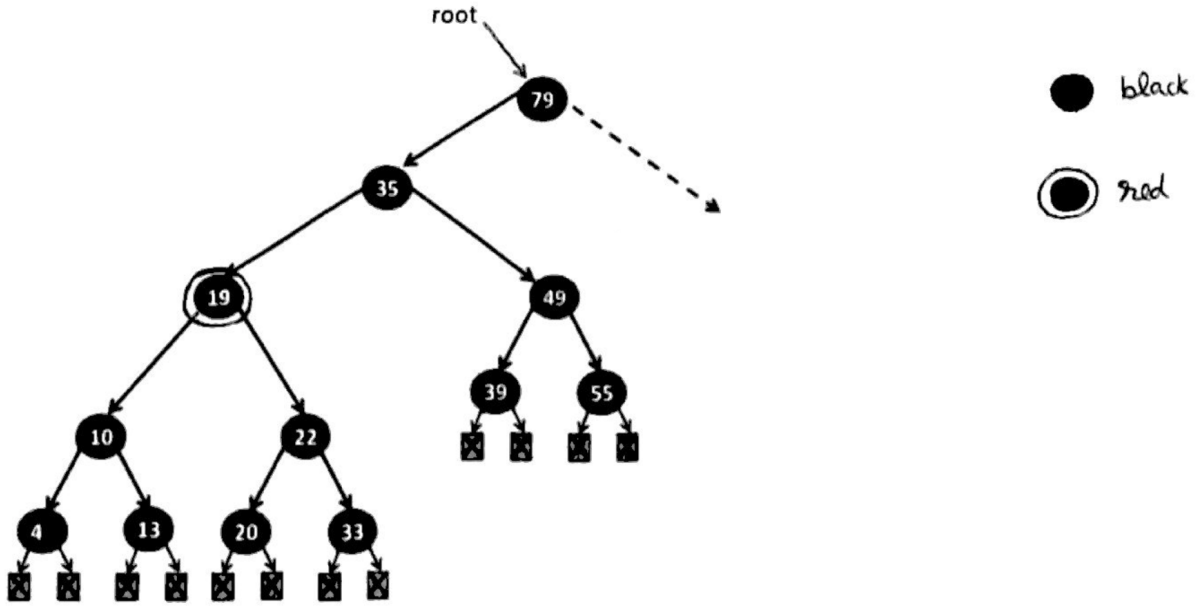

Figure 1: Q2 : Red black tree

4. (13 points) Augment the red-black tree data structure so that the following operations remain unchanged:

   - Add, delete or query a value in $O(\log n)$ time.

   In addition to this you should also give an algorithm to find the k predecessors of a given value in $O(\log n+k)$ time along with it's proof of correctness and time complexity analysis. Clearly mention the modifications made to the standard red-black tree data structure which was discussed in class. Mention if the add, delete and query operations will also have to be changed to ensure correctness of all operations and if yes, then briefly mention what changes need to be made.