CSC3102: Fall 2014
Advanced Data Structures and Algorithms Analysis

Programming Homework 1
Due: October 14, 2014

1. (40 points) A k-ary heap is like a binary heap (where $k = 2$), but (with one possible exception) non-leaf nodes have k children instead of 2 children. Implement k-ary heap as a min-priority queue for an arbitrary k 2 to support following operations:

   - `insert` (x): inserts the element x to the heap.
   - `extract-min()`: removes and returns the element of heap with the smallest key.

   k-ary heap is to be implemented using an array of predefined size. Also study the relative efficiencies of the data structure for varying values of k, by measuring the time required for performing sequence of operations given the input file for $k = 2$, 4, 6, 8, 10. Plot a bar chart. In the input file "`IN`" stands for insert and "`EX`" stand for extract-min operation. Submit bar chart and also the sample output file for the input file provided along with your code.

2. (60 points) An AVL tree is a binary tree that is height balanced: for each node x, the heights of the left and right subtrees of x differ by at most 1. Implement an AVL tree to support following operations.

   - `insert`(x): inserts an element x to the tree.
   - `search`(x): returns TRUE if a node containing element x is found in a binary tree else returns FALSE.
   - `successor`(x): returns successor of element x i.e. element with the smallest key greater than that of element x.
   - `select`(i): returns the element containing i^{th} smallest key in a binary tree.
   - `rank`(x): returns the rank/position of element x in the liner order determined by inorder traversal of tree.

   Run your program on the given input files and generate the output file. Each line in the input file is one the following commands:

   - `IN: insert`
   - `SR: search`
   - `SC: successor`

- `SE: select`
- `RA: rank`

Note:

1. This programming homework is to be done in pairs (groups of 2 students). However, both the partners should do both the problems. Spliting the load so that one person does one problem each is disallowed.

2. Either C, C++ or java can be used as a programming language. (If you choose C++ as a programming language, you can download files timer.h and timer.cpp, which are useful for timing experiments.)

3. Executing the program for a particular input file should generate output file containing result of each operation (except insert) line by line. Please look at the sample output file (contains comments after symbol // which are only for understanding purpose and need not be part of actual output file).

4. Input files attached are sample input files only. For timing experiments (Problem 1), these files need to be much bigger (having at least 10000 commands).