Indian Statistical Institute

Semester-I 2017-2018

M.Tech.(CS) - First Year

Lab Test III (6 November, 2017)

Subject: Data and File Structures Laboratory

Total: 50 marks Duration: 3 hrs.

SUBMISSION INSTRUCTIONS

1. Naming convention for your programs: `cs17xx-test3-progy.c`
2. When you have finished, copy all your files to `~dfslab/2017/labtest3/cs17xx/`.

If your program does not take input in the specified format, your code will not be evaluated, and you will get no credit.

1. *k-smallest-in-BST* (10 marks). Write a program that constructs a Binary Search Tree (BST) from a given list of non-negative integers, and then prints the k^{th} smallest element in the BST (k is a positive integer). For example, in the following BST, if $k = 3$, then output should be 28, and if $k = 6$, then output should be 50. Your program should implement at least the `insertBST(data)` function, which

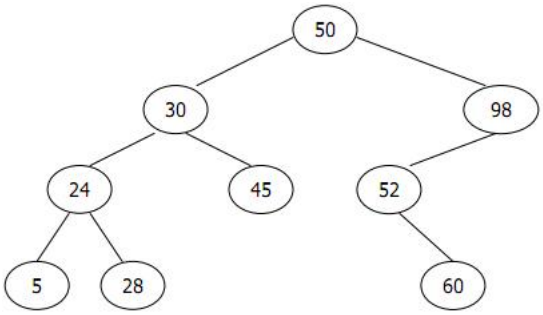

Figure 1: A BST.

should be used to insert the input integers one by one into the tree.

Input Format: The input (provided via stdin) will consist of 2 lines. The first line will contain k. The second line will consist of a list of non-negative integers separated by spaces, and terminated by -1. **The number of integers in the list will not be explicitly provided to you.** Your program should read the non-negative integers on the second line one-by-one and store them in a BST, stopping when it encounters the -1.

Output Format: Your program should print just the k^{th} smallest element of the BST.

Sample Input 0

5

5 4 3 2 1 -1

Sample Output 0

5

Sample Input 1

3

31 16 45 24 7 19 29 −1

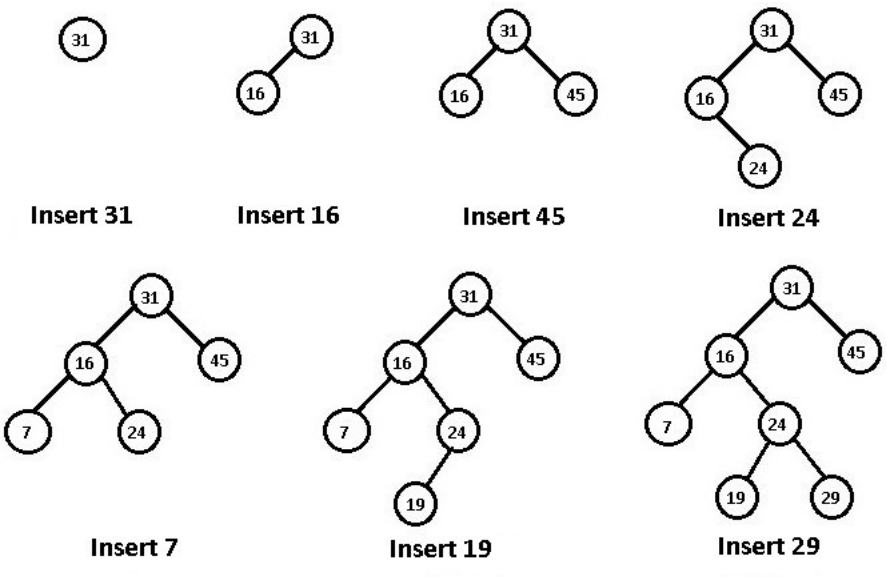

Figure 2: BST for Sample Input 0.

Sample Output 0

19

2. *isBST* (20 marks). Given a binary tree, check if it is a Binary Search Tree or not. Recall that a BST has the following properties:

   - The left subtree of a node contains only nodes with keys less than the node's key.
   - The right subtree of a node contains only nodes with keys greater than the node's key.
   - Both the left and right subtrees must also be binary search trees.

   Input Format: The first line of input will contain n, where n is the number of nodes in the binary tree. This will be followed by n more lines. Each of these remaining lines will correspond to one node in the tree and will consist of 3 integers: the key, the line number of the node corresponding to the left child (-1 if there is no left child), and the line number corresponding to the right child (-1 if there is no right child).

   Output Format: Your program should print YES or NO indicating whether the given binary tree is a BST or not respectively.

   Sample Input 0:

```
6
60   3   4
50   5   6
70   -1  7
45   -1  -1
55   -1  -1
75   -1  -1
```

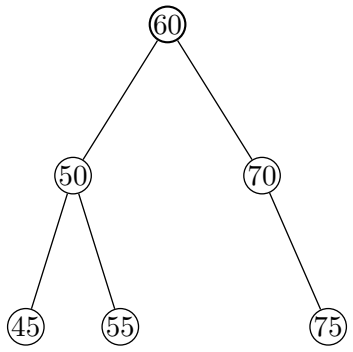

Figure 3: Binary Tree for Sample Input 0.

Sample Output 0: YES

3. *Heap Merge.* (20 marks) Given two binary MAX heaps represented using arrays, merge the given heaps into a single MAX heap.

   Input Format: The input will consist of 2 lines. Each line will contain the elements of an array representing a MAX heap. You may assume that all the heap elements are non-negative integers. The end of each array will be marked by the value -1.

   Output Format: The elements of the array representing the merged heap in a single line.

   Sample Input 0:
   10 5 6 2 -1
   12 7 9 -1

   Sample Output 0:
   12 10 9 2 5 7 6.

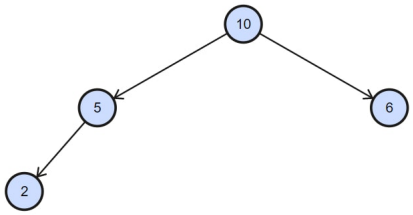

Figure 4: Heap A.

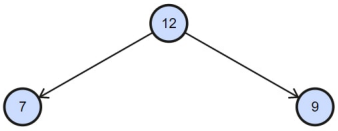

Figure 5: Heap B.

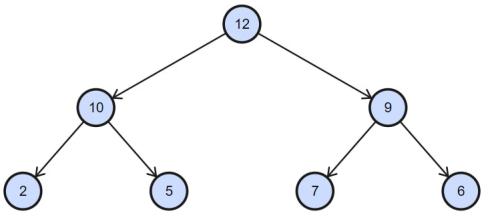

Figure 6: Merged Heap.