CpSc 2120: Algorithms and Data Structures

Instructor: Dr. Brian Dean Fall 2019
Webpage: `http://www.cs.clemson.edu/~bcdean/` MW 2:30-3:45
Handout 8: Quiz 1 Coding Part. 30 possible points. Daniel 313

You may use any resources associated with this class for this quiz (e.g., pre-existing notes or code you have written for this class, code or notes from the class Canvas site). You can look at man pages and `cppreference.com` for general C/C++ reference, but you may not consult any other material from the web. Feel welcome to use prior code from the Canvas site in your solutions if you feel this is helpful. You must upload your solutions to handin by the end of your lab session; files with upload timestamps beyond the end of lab will not be counted.

1 Unique Merge (8 points)

Given two linked lists A and B that are in sorted order, a *unique merge* operation creates a single newly-allocated linked list in sorted order containing all of the distinct elements in A and B. Please implement this operation efficiently. A recursive solution will receive at most 7/8 points (since it would potentially crash on large inputs due to limited stack space). You can download starter code for this part by typing

```
scp ada3:/tmp/umerge.cpp .
```

2 Pre-Select (11 points)

Bored with in-order traversals, Dhruv has recently become a fan of pre-order traversals of binary search trees[1]. Recall that the pre-order traversal of a tree is generated by first printing the root, and then recursively printing a pre-order traversal of its left and right subtrees:

```
void preorder(Node *root)
{
   if (root == NULL) return;
   cout << root->key << "\n";
   preorder(root->left);
   preorder(root->right);
}
```

[1]For one nice property: the pre-order traversal of a tree uniquely specifies not only the elements in the tree, but also the "shape" of the tree, while an in-order traversal only gives the elements in the tree but not necessarily the "shape" — many different tree shapes could generate the same in-order traversal.

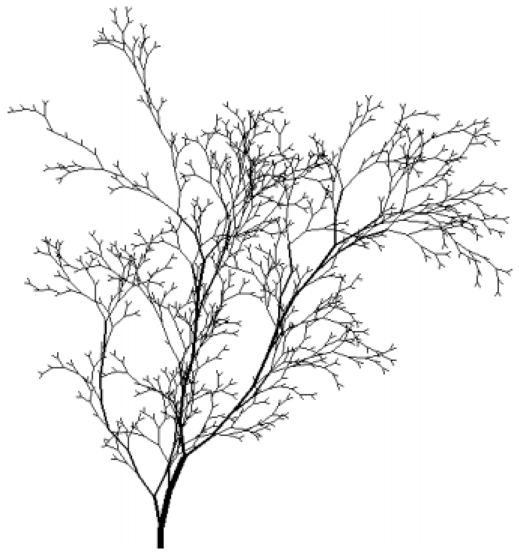

Figure 1: An algorithmically generated picture of a tree, from the book "The Algorithmic Beauty of Plants" by Prusinkiewicz and Lindenmayer.

You are given as input a length-n array containing the pre-order traversal of a balanced binary search tree containing n distinct elements. Please build an efficient implementation of the *select*(r) operation, which returns the rth largest integer in the array (e.g., $r = 0$ for the minimum, $r = n-1$ for the maximum).

You can download starter code for this part by typing

```
scp ada3:/tmp/preselect.cpp .
```

3 Twin Trees (11 points)

The Gurrie twins are walking in the Clemson experimental forest one day, wondering if any of the nearby trees have twins[2]. After searching for a while and not finding any twin trees, the Gurries decide to search for twins of another type of tree: the binary search tree. They have a list of n binary search trees, each containing n nodes (in our input, we'll use $n = 10,000$). Exactly two of these trees are identical. All other trees differ in some way, either in terms of the keys stored at some of their nodes, or in terms of the "shape" of the tree. Please write a program to help identify which two trees are the identical ones. You can assume the trees are all reasonably balanced.

You can download starter code for this part by typing

```
scp ada3:/tmp/twintree.cpp .
```

[2] Apparently trees can have twins, just like people.