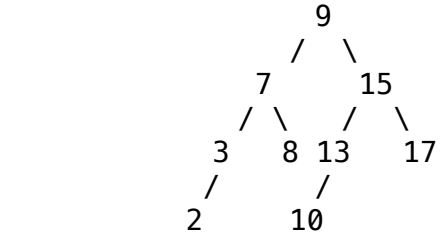
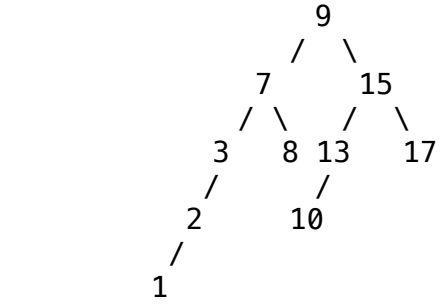
CS 010C – Assignment 3
Andrew Pham
Collaborated with Arjun Bhalla, Rushil Shah, and Devesh Panda

Problem 1: (10 points) For the binary search tree (BST) shown below, carry out the operations shown below in the given order. The state of the tree after each operation carry out to the next operation. For example, after you insert the value 10 in part (a), assume that the value stays there for all the following parts. In all operations, assume we use the basic insert or delete algorithms of a BST unless otherwise noted.

1. (2 point) Insert the value 10 in the tree.



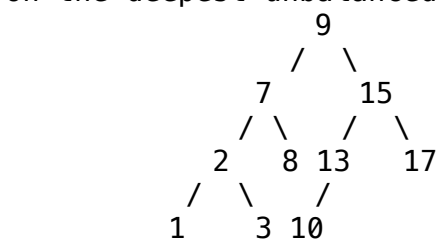
2. (2 point) Insert the value 1 in the tree.



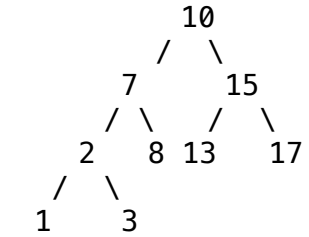
3. (2 point) According to the AVL tree balancing condition, is the tree currently balanced? List down all the unbalanced tree nodes?

The tree is NOT currently balanced. Node containing 3 has balance factor of 2, Node containing 7 has balance factor of 2.

4. (2 point) Make the tree balanced by making the appropriate rotation on the deepest unbalanced node.



5. (2 point) Delete the value 9 from the tree. You do not need to balance the tree after deletion.



Problem 2: (10 points) For the array given below, show the state of the array after running each of the given sorting algorithms as described below. Assume that the sorting algorithms are designed to sort the array in ascending order.

[8, 7, 9, 10, 4, 3, 1, 5, 6, 2]

1. (4 points) Selection sort. Show the state of the array after running one iteration of the outer loop.

[1, 7, 9, 10, 4, 3, 8, 5, 6, 2]

2. (3 points) Quick-sort algorithm. Assume that we use the median-of-three pivot selection technique. Show the state of the array after the first partition step.

Our pivot will be 4, found by getting the median of the first, middle, and last elements of the array (8, 4 and 2). As a preprocessing step specified by the quick sort algorithm, we will swap 4's position with that of the first element, 8.
[2, 3, 1, 4, 8, 7, 9, 5, 6, 10]

3. (3 points) Quick-sort algorithm. Assume that we use the first element pivot selection technique. Show the state of the array after the first partition step.

[2, 7, 4, 3, 1, 5, 6, 8, 9, 10]

Problem 3:
1. (5 points) Write your algorithm based on Selection Sort

```
bool C(string String1, string String2) {
    return String1 < String2;
}
int* Index_Sort_Selection(string A[], int n) {
    if(n > 0) {
        int *I = new int(n);
        for(int x = 0; x < n; x++) {
            I[x] = x;
        }
        for(int i = 0; i < n; i++) {
            int min = i;
            for(int j = i + 1; j < n; j++) {
                if(C(A[j], A[min])) {
                    int temp = I[min];
                    I[min] = I[j];
                    I[j] = temp;
                }
            }
        }
        return I;
    }
    return nullptr; //if invalid n
}
```

2. (5 points) Write your algorithm based on Quick Sort

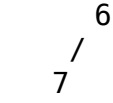
```
bool C(string String1, string String2) {
    return String1 < String2;
}
int* Index_Sort_Quick(string A[], int n) {
    if(n > 0) {
        int *I = new int(n);
        for(int j = 0; j < n; j++) {
            I[j] = j;
        }
        Index_Quicksort(A, I, 0, n);
        return I;
    }
    return nullptr; //if invalid n
}
void Index_Quicksort(string A[], int I[], int left, int right) {
    if(left < right) {
        int q = Index_Partition(A, I, left, right);
        Index_Quicksort(A, I, left, q-1);
        Index_Quicksort(A, I, q+1, right);
    }
}
```

```
int Index_Partition(string A[], int I[], int left, int right) {
    string p = A[left];
    int i = left + 1;
    for(int j = left + 1; j < right; j++) {
        if(C(A[j], p)) {
            int temp = I[j];
            I[j] = I[i];
            I[i] = temp;
            i++;
        }
    }
    int temp = I[left];
    I[left] = I[i-1];
    I[i-1] = temp;
    return i-1;
}
```

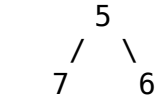
Problem 4: (10 points) Insert the following values into an, initially empty, min heap in the given order. Show the state of the heap after each insertion.

Insert 6:

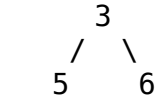
Insert 7:



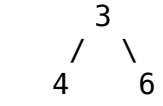
Insert 5:



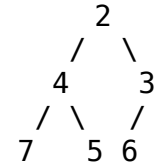
Insert 3:



Insert 4:



Insert 2:



Insert 1: