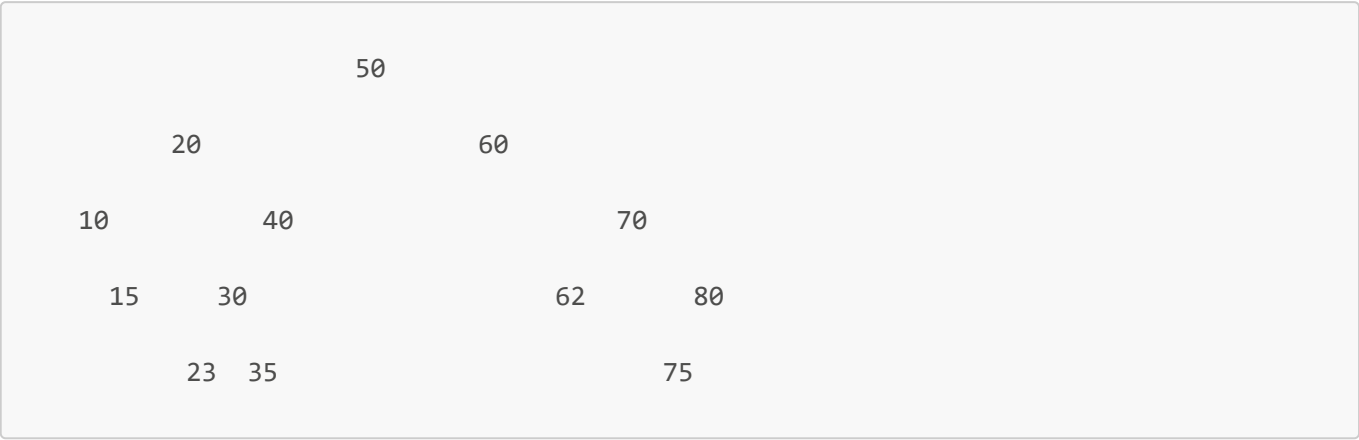


Homework 5 Solution

Problem 1:

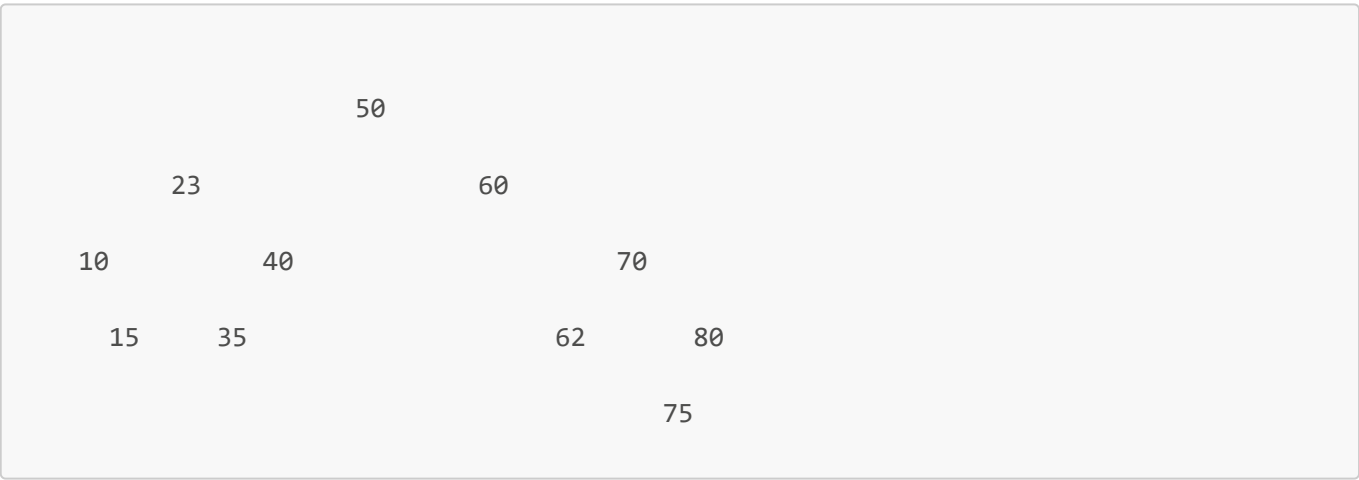


In-order: 10 15 20 23 30 35 40 50 60 62 70 75 80

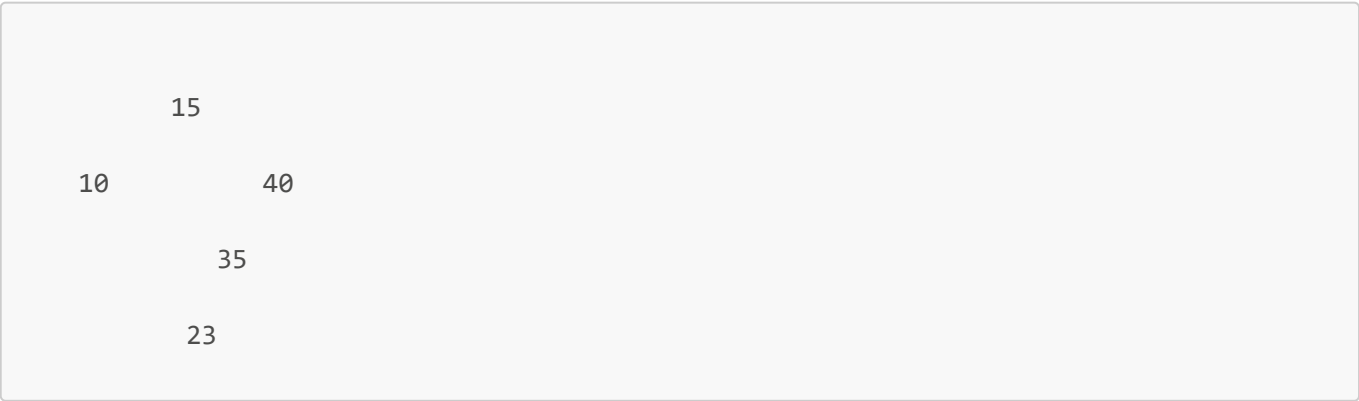
Pre-order: 50 20 10 15 40 30 23 35 60 70 62 80 75

Post-order: 15 10 23 35 30 40 20 62 75 80 70 60 50

One possibility is



Other possibilities have the left subtree of 50 being



or

```

      15
    10    40
      23
      35
  
```

Problem 2:

```

struct Node
{
    int data;
    Node* left;
    Node* right;
    Node* parent;
};

void insertAuxiliary(Node*& n, int value, Node* par)
{
    if (n == nullptr)
        set n to point to a new Node whose data field is set to value,
        whose left and right children are null, and whose parent field
        is set to par.
    else if (value < n->data)
        insertAuxiliary(n->left, value, n);
    else
        insertAuxiliary(n->right, value, n);
}

void insert(Node*& n, int value)
{
    insertAuxiliary(n, value, nullptr); // pass nullptr as parent of root
}
  
```

Problem 3:

```

      7
    5   6
  4 0   3
  
```

7 5 6 4 0 3

6 5 3 4 0

Problem 4:

$O(C + S)$. We'd have to do a linear search through the outer vector to find the course, which is $O(C)$, and then after that do a linear search of the S students in the list, which is $O(S)$. We can't do a binary search in a linked list in logarithmic time, because it takes linear time just to get to the middle item of a list, sorted or not.

$O((\log C) + S)$. A BST-based map finds the course in $O(\log C)$ time, and after that we do a linear search of the S students in the list.

$O(\log C + \log S)$. A BST-based map finds the course in $O(\log C)$ time, and a BST-based set finds the student in $O(\log S)$ time. Notice that a mathematically equivalent way to write this is $O(\log CS)$, but that form makes it harder to understand why it's right.

$O(\log S)$. A hash-based map finds the course in $O(1)$ time, and a BST-based set finds the student in $O(\log S)$ time. Notice that a constant like 1 is dominated by a function that grows with S , so we write $O(\log S)$ instead of $O(1 + \log S)$.

$O(1)$. A hash-based map finds the course in $O(1)$ time, and a hash-based set finds the student in $O(1)$ time.

$O((\log C) + S)$. A BST-based map finds the course in $O(\log C)$ time, and we can visit all S items in order in a BST-based set in $O(S)$ time.

$O(S \log S)$. A hash-based map finds the course in $O(1)$ time, and although we can visit all S items in a hash-based set in $O(S)$ time, they'd be in whatever order the hash function caused them to be scattered across the buckets. We have to sort them. It takes $O(S)$ steps to get them into and out of an auxiliary vector, say, and $O(S \log S)$ to sort them.

$O(C \log S)$. The hash-based map is keyed on the course, not the student, so it's not organized to look up students efficiently. Our only choice is to check every course and see in $O(\log S)$ time whether the student is in its set of students. Since we have to do the $O(\log S)$ operation for each of the C courses, it's $O(C \log S)$.