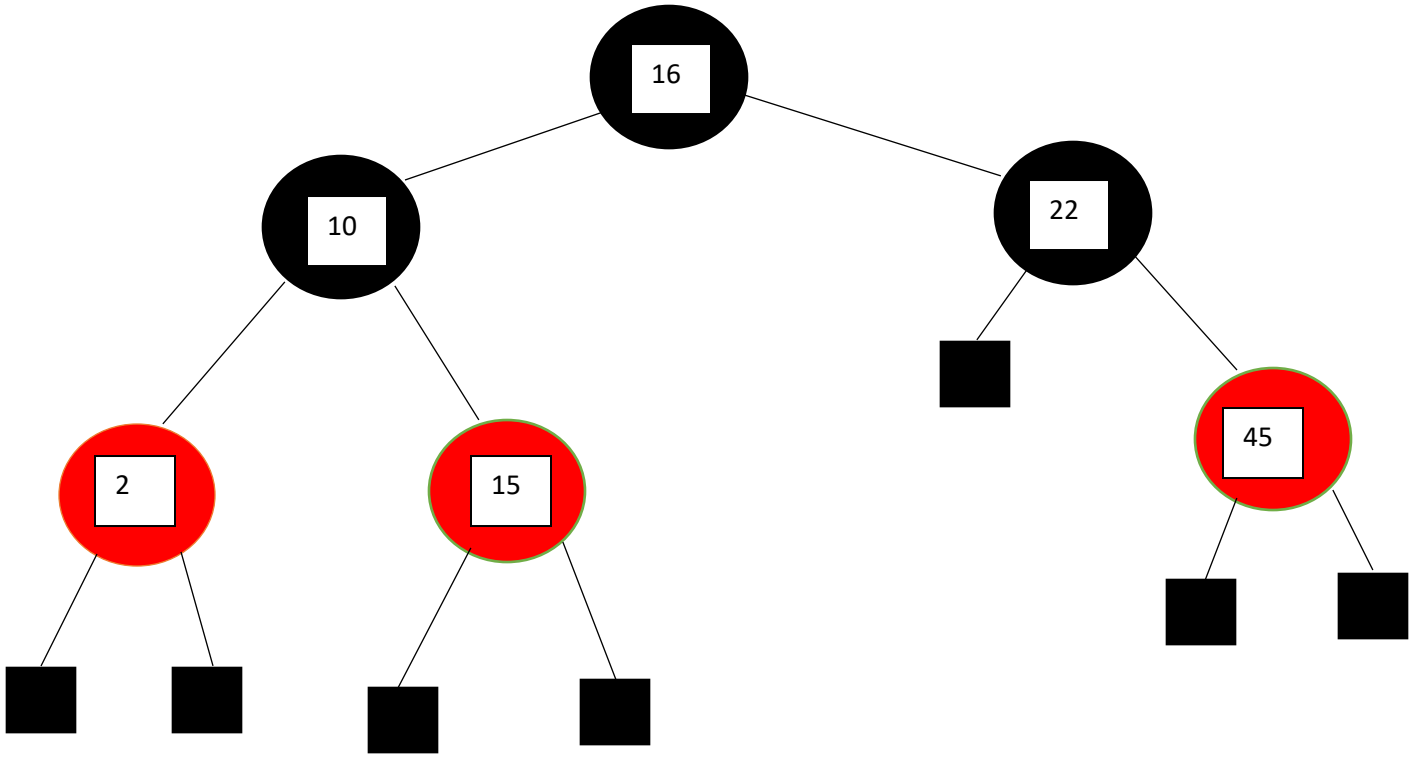
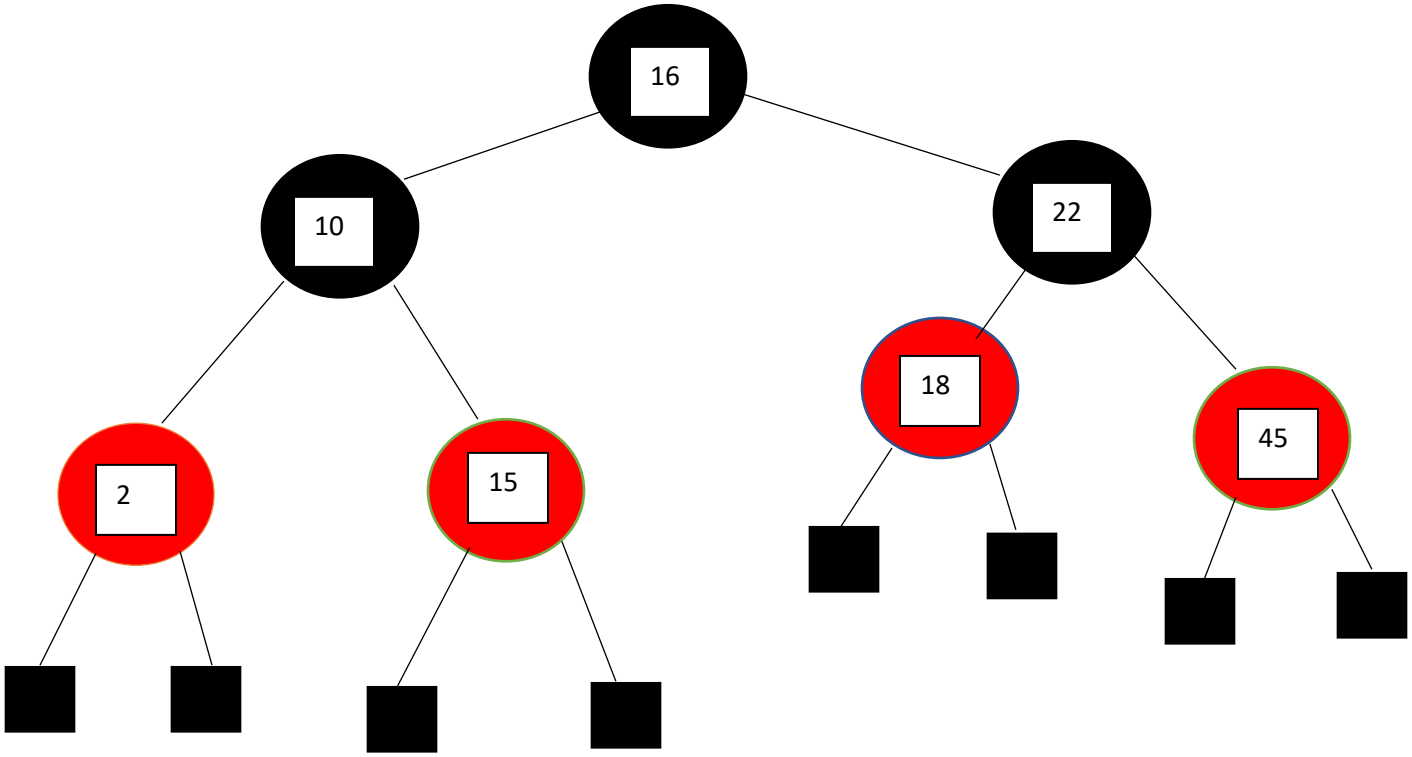
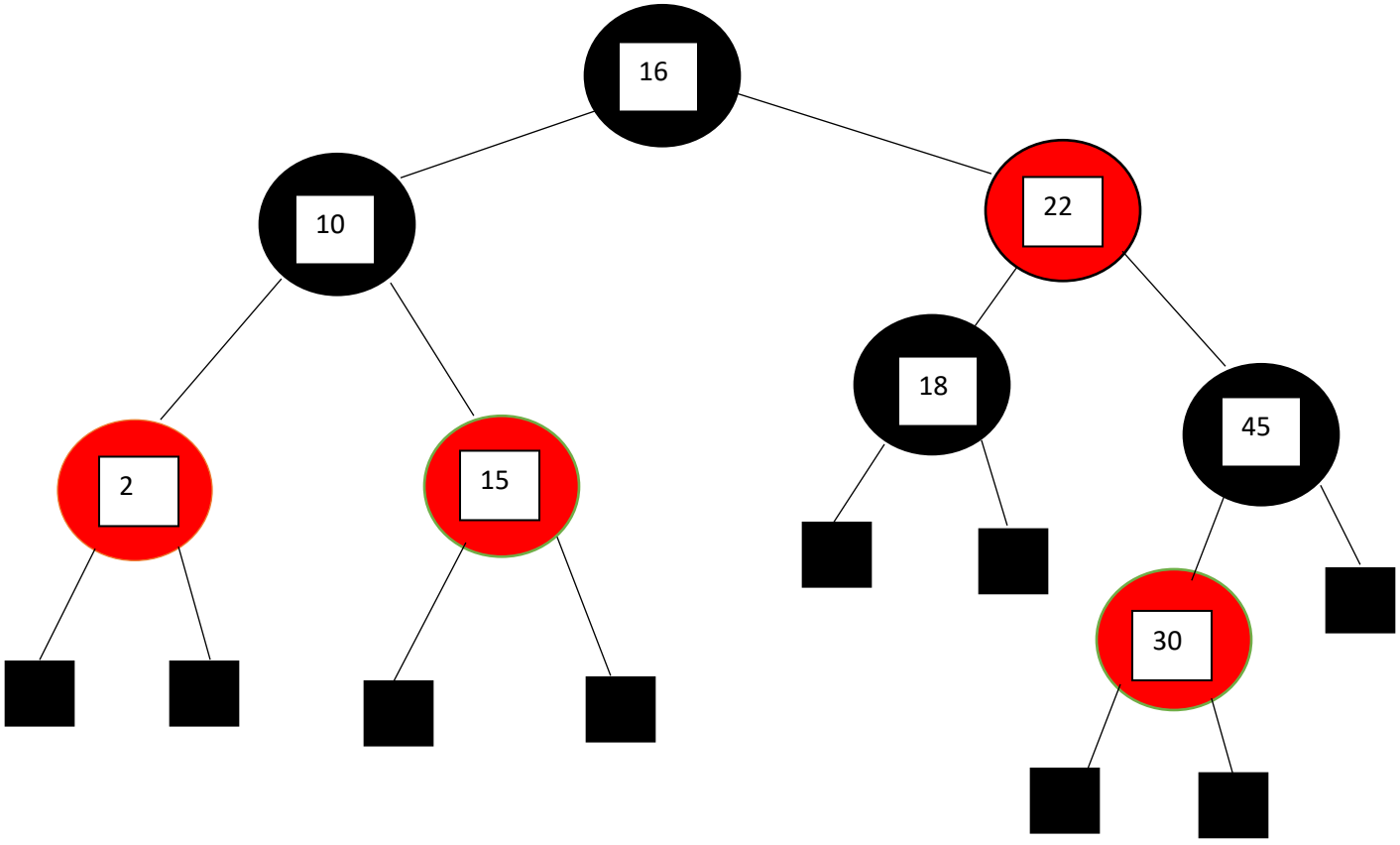
Chris Grimes
CS 46101
Homework #4

1. (5 points) Consider the following sequence of keys: (18, 30, 50, 12, 1). Insert the items with this set of keys in the order given into the red-black tree in the figure below. Draw the tree after each insertion.

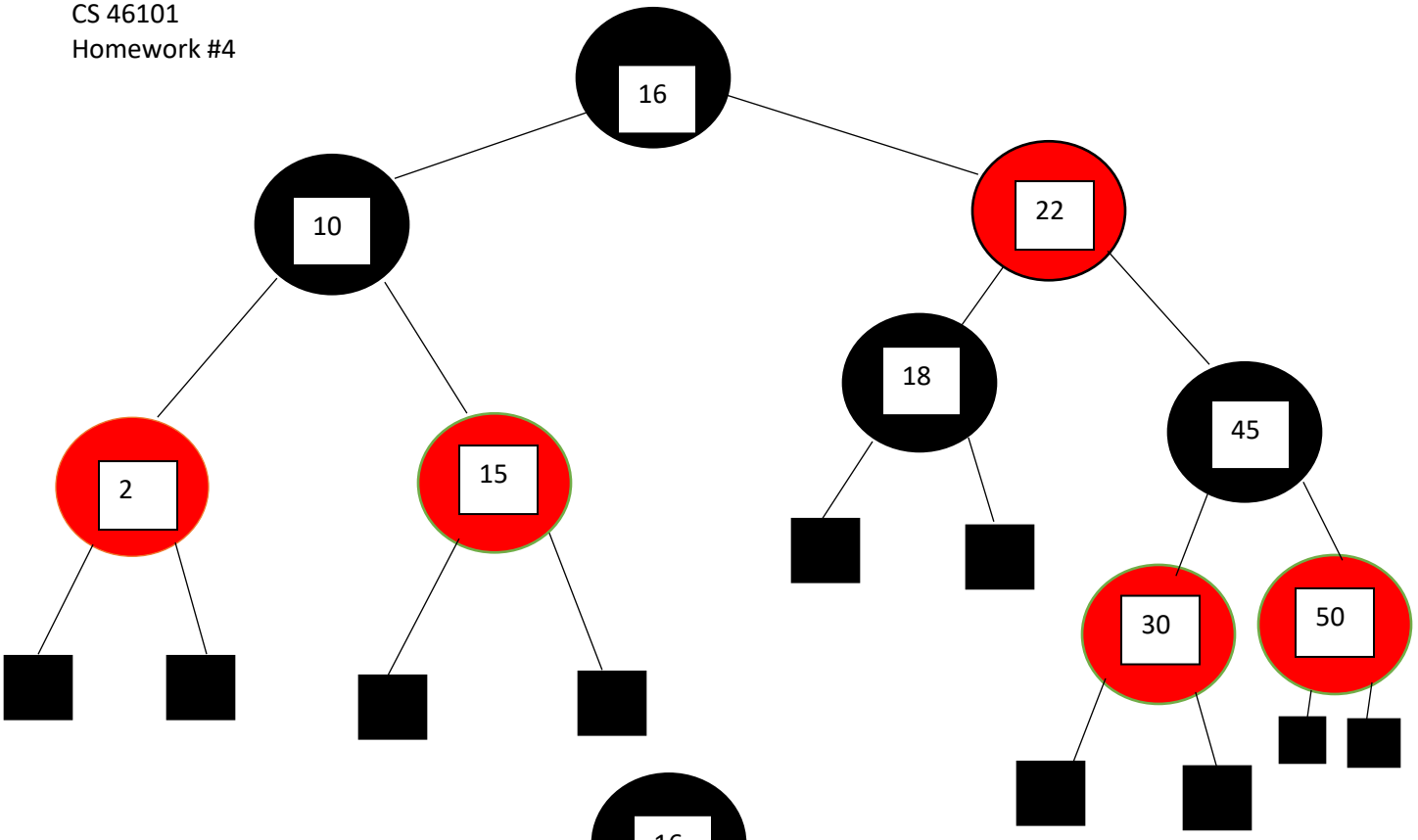


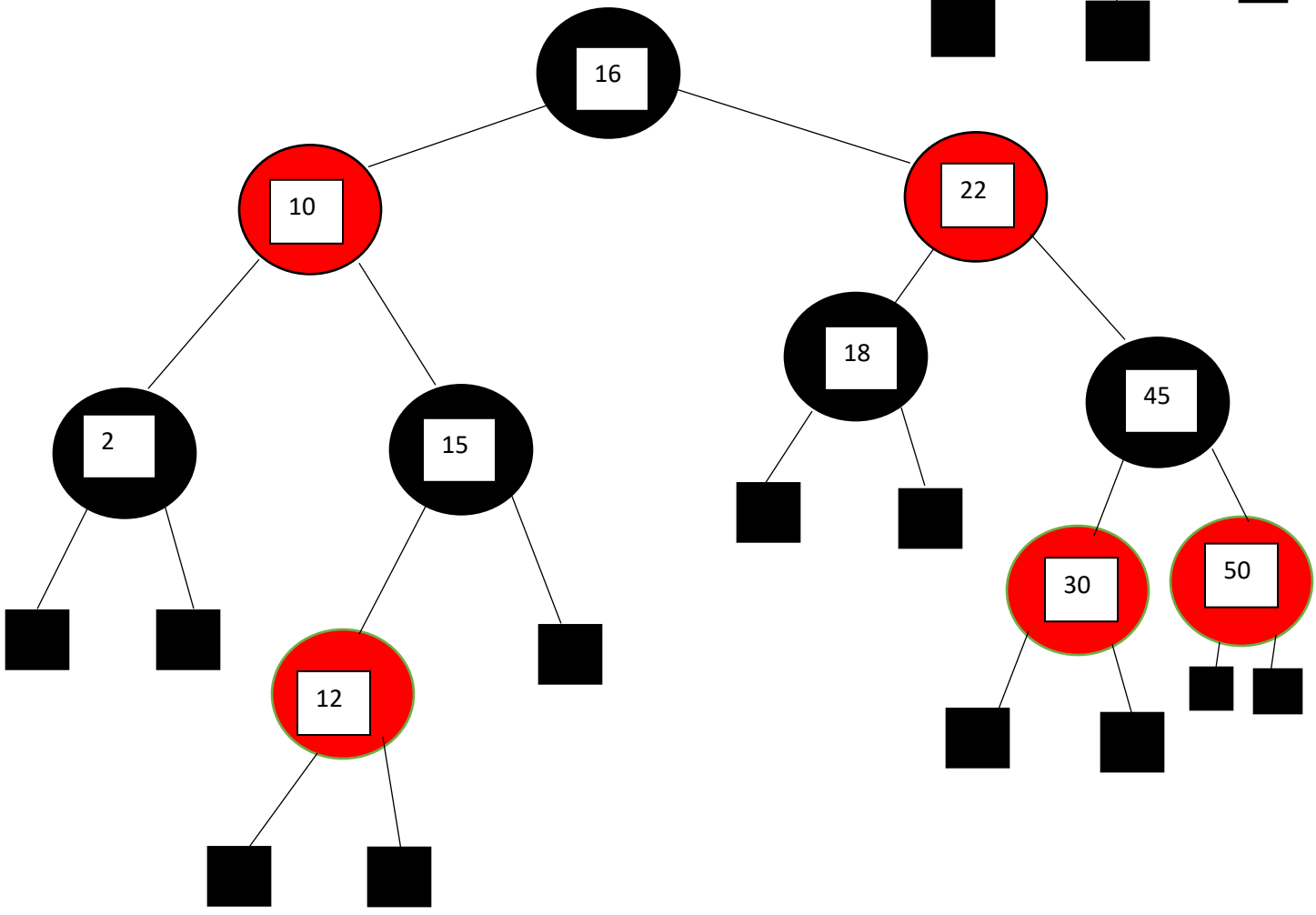
Chris Grimes
CS 46101
Homework #4



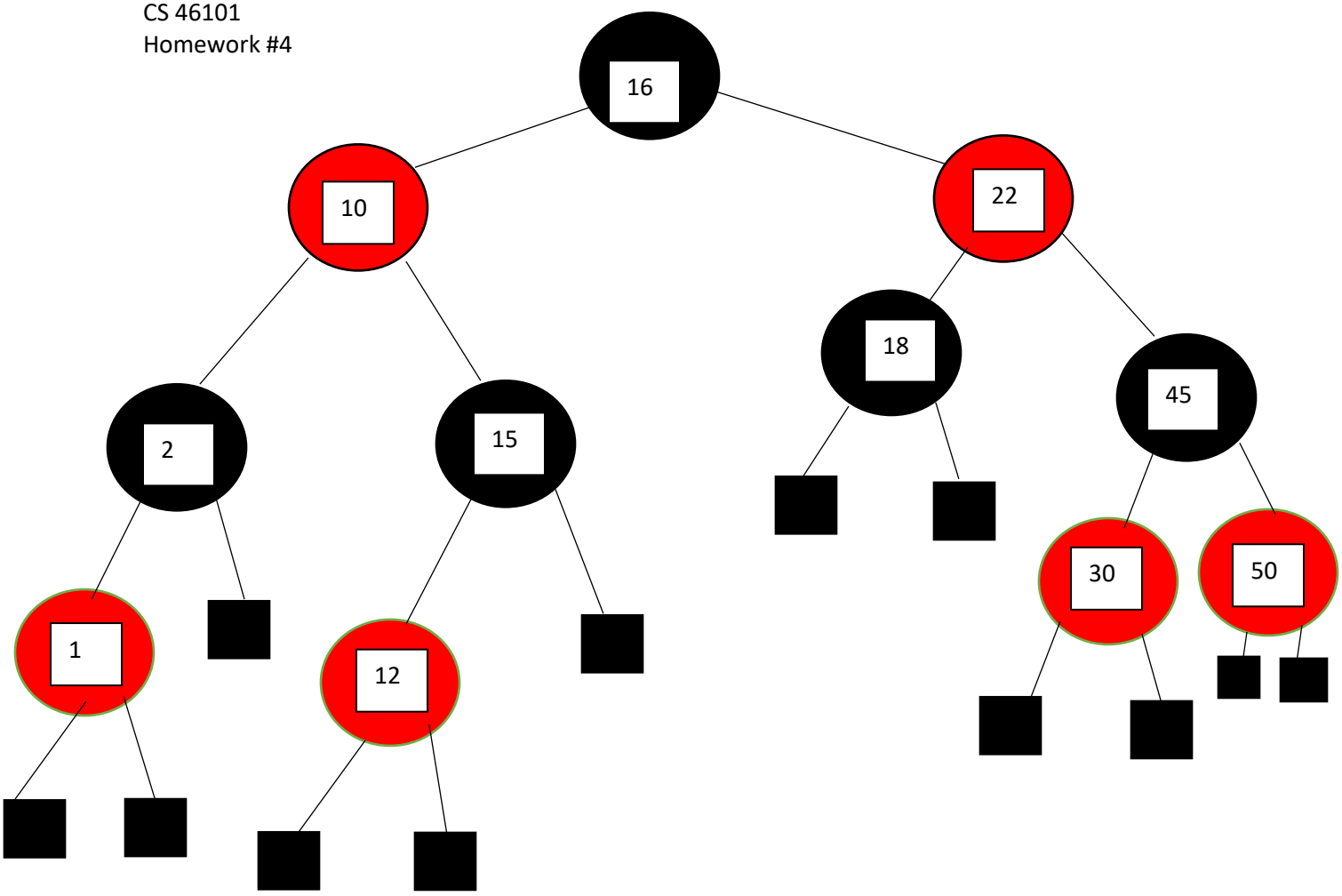


Chris Grimes
CS 46101
Homework #4





Chris Grimes
CS 46101
Homework #4



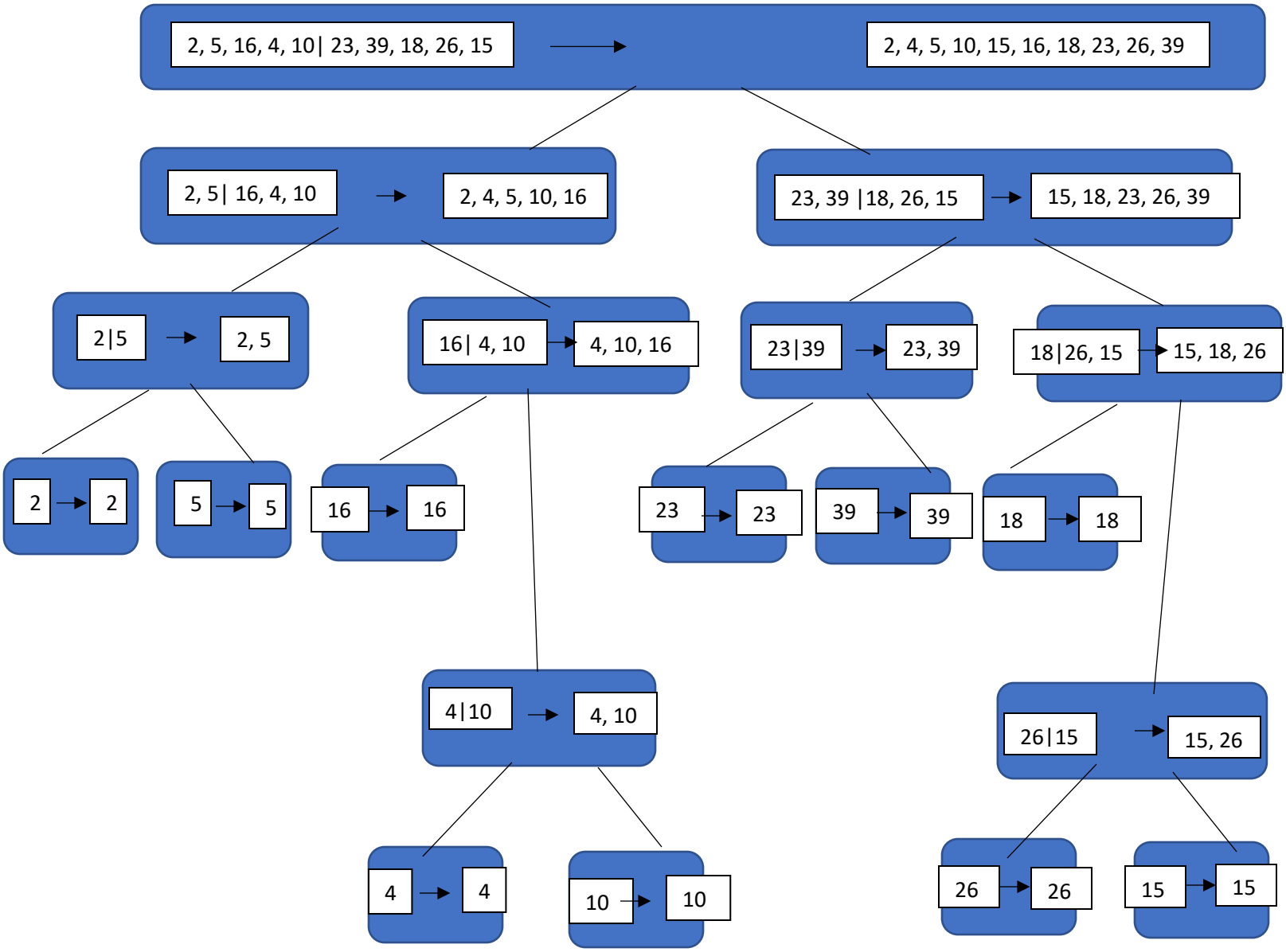
Chris Grimes
CS 46101
Homework #4

2. (10 points) Design and give the pseudocode for an O(log n) algorithm that determines whether a red-black tree with n keys stores any keys within a certain (closed) interval. That is, the input to the algorithm is a red-black tree T and two keys, l and r, where $l \leq r$. If T has at least one key k such that $l \leq k \leq r$, then the algorithm returns true, otherwise it returns false. Hint: You can use the TreeSearch algorithm (page 146) as a subroutine.

```
Algorithm findRange(T, l, r)
        Input: a tree T, a lower value l, an upper value r
        Output: Boolean value true if said tree has at least one key between the values l and r
        if( root <= r && root>=l)
                return true
        else if (root has a left child)
                if(root>=r)
                        findRange(left child, l, r)
        else if (root has a right child)
                if(root <=l)
                        findRange(left child, l, r)
        else
                return false
```

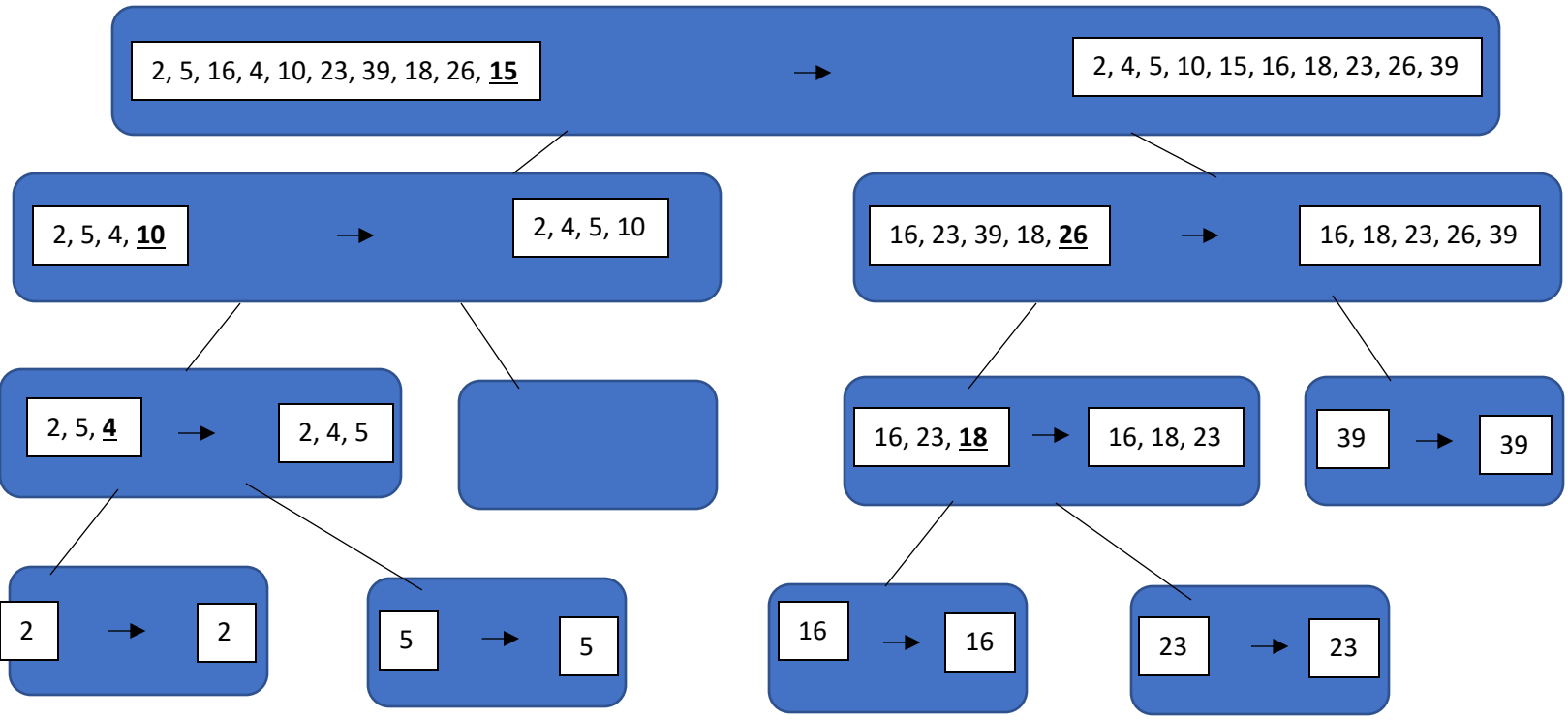
Chris Grimes
CS 46101
Homework #4

3. (a) (5 points) Draw the merge-sort tree for an execution of the merge-sort algorithm on the input sequence: (2, 5, 16, 4, 10, 23, 39, 18, 26, 15) (like in Figure. 4.2).



Chris Grimes
CS 46101
Homework #4

(b) (5 points) Draw the quick-sort tree for an execution of the quick-sort algorithm on the input sequence from part (a) (2, 5, 16, 4, 10, 23, 39, 18, 26, 15) (like in Figure 4.12). Use the last element as the pivot.



Chris Grimes
CS 46101
Homework #4
(c) (3 points) What is the running time of the version of quick-sort that uses the element at rank floor of n/2 as the pivot, provided that the input sequence is already sorted? Explain.

If we are given a sorted sequence and we are to use the floor of n/2 as the pivot for quick-sort the expected run time should be O(n*log(n)) because our pivot will always give us two sequences of similar size and thus we will never have an empty, or nearly empty, sequence and a sequence with ¾ more elements.

Chris Grimes
CS 46101
Homework #4

4. (10 points) Suppose we are given a sequence S of n elements, each of which is colored red or blue. Assuming S is represented by an array, give a linear-time in-place algorithm for ordering S so that all the blue elements are listed before all the red elements. What is the running time of your method?

```
Algorithm sortRvsB(S)
    Input: a sequence S
    Output: a sequence in which all the blue elements are listed before the red elements
    pointer low<- S[0]
    pointer high<- S[size of S -1}
    while(low <high && low< the last element of S)
        while(low != red element)
            ++low
        while(low=red element)
            if(low=red element && high=blue element && low <high)
                swap(low, high)
            else if(low< high)
                --high
    return S
```

The worst case running time of this algorithm is O(n) where n is the number of elements.

Chris Grimes
CS 46101
Homework #4

5. (10 points) Let A and B be two sequences of n integers each. Give an integer m, describe an O(n log n) time algorithm for determining if there is an integer a in A and an integer b in B such that m = a + b.

```
Algorithm findSum(A, B, m)

        Input: an integer m and two sequences A and B

        Output: return 1 if there is one integer from each sequence that sums to m, if no pair is found return -1

        Sort(sequence A in ascending order)

        Sort(sequence B in desending order)

        pointer a<-A[0]

        pointer b<- B[0]

        while(a <= the last element in A && b<= the last element in B)

                if(*a+*b>m)

                        ++b

                else if(*a+*b<m)

                        ++a

                else if(*a+*b=m)

                        return 1

        return -1
```