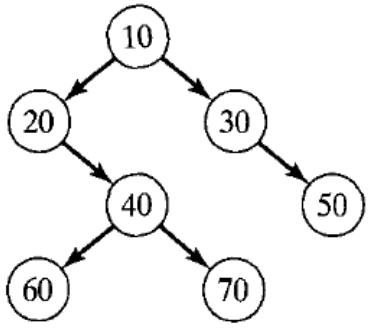
CST 370 – Spring A 2020
Homework 1

Name: Sara Kazemi

Class ID: 4444

1. (10 points) Is the following tree a binary tree? If so, is it a binary search tree?
Yes, the following tree **is a binary tree** since each vertex has no more than two children, and each child is a left or right child of a given parent. However, it **is not a binary search tree** because the number assigned to each parental vertex is not larger than the numbers in the left subtree. To be considered a binary search tree, the numbers in the left subtree should all be less than the number in the parent node, while the numbers in the right subtree should all be greater than the number in the parent node.



2. (10 points) (a) Find **gcd(270, 192)** by applying Euclid's algorithm as we covered in the class. Present the **intermediate steps** of the algorithm clearly.

$$gcd(270, 192) = gcd(192, 78) = gcd(78, 36) = gcd(36, 6) = gcd(6, 0) = \mathbf{6}$$

(b) Find **gcd(270, 192)** by applying the middle school approach (= prime factors) as we covered in the class. Present the **intermediate steps** of the approach clearly.

$$192 = 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 3$$
$$270 = 2 \times 3 \times 3 \times 3 \times 5$$

So,

$$gcd(270, 192) = 2 \times 3 = \mathbf{6}$$

3. (15 points) Consider the following pseudocode. **Describe what this algorithm computes.**
Ultimately, the algorithm subtracts the smallest value in the array from the largest value in the array; so, this algorithm computes and returns the difference between the largest value and the smallest value in the array.

```
Algorithm Compute(A[0.. n – 1])
1.  num1 ← A[0];
2.  num2 ← A[0];
3.  i ← 1
4.  while i < n do
5.      if A[i] < num1
6.          num1 ← A[i];
7.      if A[i] > num2
8.          num2 ← A[i];
9.      i ← i + 1
10. return (num2 – num1);
```

4. (15 points) Consider the following algorithm to sort an array:

```
Algorithm ComparisonCountingSort(A[0..n − 1], S[0..n − 1])
//Sorts an array by comparison counting
//Input: Array A[0..n − 1] of orderable values
//Output: Array S[0..n − 1] of A's elements sorted in nondecreasing order
for i ← 0 to n − 1 do
    Count[i] ← 0
for i ← 0 to n − 2 do
    for j ← i + 1 to n − 1 do
        if A[i] < A[j]
            Count[j] ← Count[j] + 1
        else Count[i] ← Count[i] + 1
for i ← 0 to n − 1 do
    S[Count[i]] ← A[i]
```

(a) Apply the algorithm to sort the list 40, 50, 10. In other words, assume that the array **A[ ]** has the initial values 40, 50, and 10. Present the contents of the arrays **A[ ]**, **S[ ]**, and **Count [ ]** after finishing the execution of the algorithm.
A = [40, 50, 10]
S = [10, 40, 50]
Count = [1, 2, 0]

(b) Similarly, present the contents of the three arrays if the array **A[ ]** has the initial values 60, 10, 60, 10, 20, 50?
A = [60, 10, 60, 10, 20, 50]
S = [10, 20, 20, 50, 60, 60]
Count = [5, 1, 4, 0, 2, 3]

5. (50 points) Write a C++ program called **hw1_1.cpp** that reads an input file name from a user. Note that the input file contains several integer numbers. After reading the numbers from the file, your program should display the closest distance between two numbers among the numbers. For example, if the input numbers are -1, 10, 3, -15, 7, 20, 150, 200, 50, and 4, the answer should be 1 because the distance between 3 and 4 is 1. Note that the distance between two numbers is always positive.

See attached cpp file.