Data Structure and Algorithms
Assignment 4

Deadline: Tuesday, July 09, 2024, 11:59 PM

Question 1:

Given a Binary Search Tree (BST), do the following:

a) Write a function to determine if a binary tree is a valid BST. The function should return true if the tree satisfies the BST properties and false otherwise.

b) Write a function to convert the BST into a sorted array in ascending order.

c) Write a function to find the distance between two nodes in a BST. The distance is defined as the number of edges between the two nodes in the tree.

d) Given a two binary search tree, write a function to determine if two trees are identical or not. Note: Two trees are identical when they have the same data and the arrangement of data is also the same.

e) Write a function to determine if a BST is balanced. A balanced BST is defined as a tree in which the heights of the left and right subtrees of any node differ by at most 1.

f) Calculate the height of the tree.

Question 2:

Sort the values present in the file "input.txt" in ascending and descending order using min and max heaps.

Question 3:

Using `unordered_map`, read multiple strings from the console as input from the user. The input ends when the user enters "`stop`". At the end, display all the input strings along with their counts.

Question 4:

Suppose you have a **doubly circular linked list** of integers with a **`tail`** pointer. You are required to implement a function that should reverse every consecutive group of `k` nodes in the list:

```
void reverseInGroups (int k)
```

For example (assuming the values below are placed in eleven different nodes):

- Input: `[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]` and `k = 3`
- Output: `[3, 2, 1, 6, 5, 4, 9, 8, 7, 11, 10]`

Note: If the number of nodes in the last group is less than `k`, they should also be reversed (see 10, 11).

Practice question – submission is not required:

Provide a stack trace of the following code, you need to draw the stack trace on paper.

```
void secretOperation(int n)
{
    if (n > 3)
        secretOperation(n - 1);
    for (int i = 3; i <= n; i++)
        cout << (n+3) % i;
    cout << endl;
}

int main()
{
    secretOperation(7);
    return 0;
}
```