**Problems in Data Structures and Algorithms – Final Exam**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name**: | Yusuf Furkan Yücesoy | **ID:** | 152120151005 | **Section:** |  |

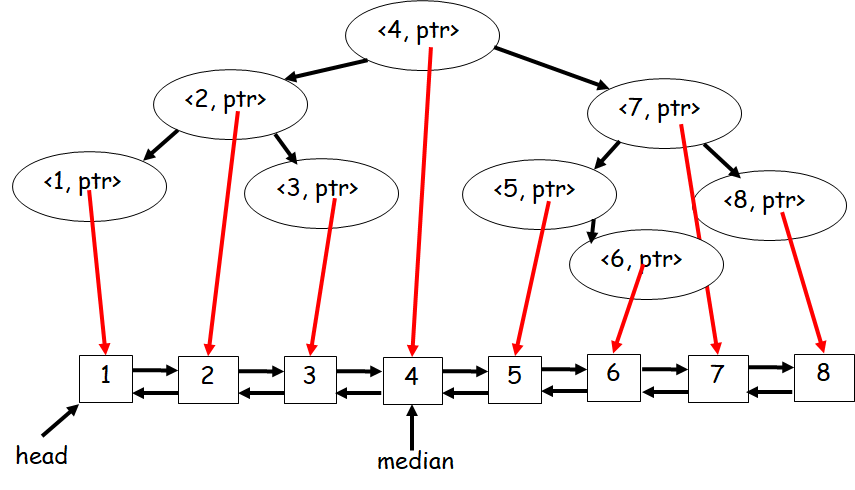
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Q1** | **Q2** | **Q3** | **Q4** | **Q5** | **Total** |
|  |  |  |  |  |  |

**Q1. Finding Medians Efficiently (20 points)**

You have a randomly generated dynamic (expanding) list of numbers. You would like to store these numbers in a DS that not supports

* Insert operations in O(logN)
* Median queries in O(1) time
* Delete operations in O(logN).

Here is how you would implement this Data Structure:



Fill in the data structure methods given in **Q1.h** and **Q1.cpp**.

**Q2. Pair Sum (20 points)**

Given a sorted array of distinct integer values, find all pairs of numbers (x, y) s.t. x < y and (x+y)%10 = k, 0<=k<=9. Your algorithm must return the number of such pairs, not the pairs themselves and run in O(n) time. For example:

A=[1, 2, 3, 4, 7, 9, 10, 11, 12, 13, 20, 23, 30], k=3

Pairs: (1, 2), (1, 12), (2, 11), (3, 10), (3, 20), (3, 30), (4, 9), (10, 13), (10, 23), (11, 12), (13, 20), (13, 30), (20, 23), (23, 30)

# of pairs: 14.

Fill in the function “int NoOfPairs(vector<int>& A, int k)” given in **Q2.cpp**.

**Q3. Sort Stack (20 points)**

You are given a stack of elements, and are asked to sort this stack in increasing order so that the smallest element is at the top, and the largest element is at the bottom of the stack. You only have the regular stack operations: push, pop, top, isEmpty. You may make use of an additional stack BUT are NOT allowed to copy all stack elements into an array, sort the array and copy the elements back to the stack.

Your algorithm must have a worst case running time performance of O(N2), and a best case running time of O(N). That is, if the stack is already sorted, it must run in O(N).

**Hint**: Implement an insertion sort algorithm using two stacks.

Fill in the function “void SortStack(stack<int>& S)” given in **Q3.cpp**.

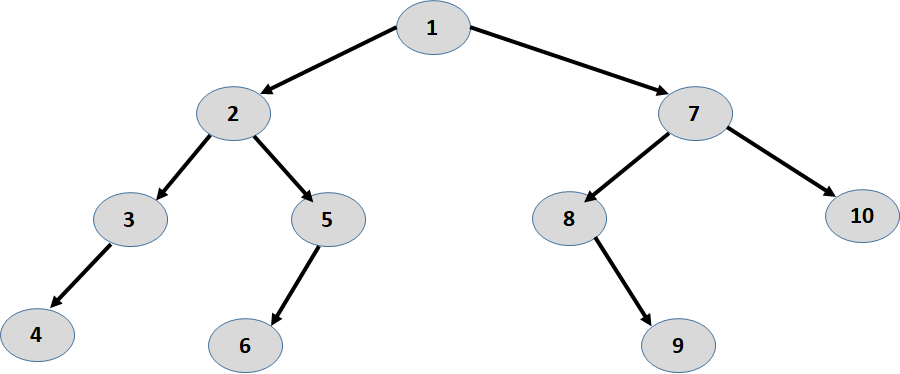
**Q4. Construct Binary Tree from Pre-Order and In-Order Sequences (20 points)**

You are given two sequences: One that corresponds to the pre-order traversal of a binary tree. Another sequence that corresponds to the in-order traversal of the same binary tree. You are asked to implement an algorithm that constructs the binary tree given these two sequences. For example, given the following two sequences:

preOrderSeq = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

inOrderSeq = { 4, 3, 2, 6, 5, 1, 8, 9, 7, 10}

The constructed tree will look like:



Fill in the function “BinaryTreeNode\* ConstructTree(vector<int>& preOrderSeq, vector<int>& inOrderSeq)” given in **Q4.cpp**.

**Q5. Minimum in a Rotated Sorted Array (20 points)**

Suppose an array sorted in ascending order is rotated at some pivot unknown to you beforehand. (i.e., [0, 1, 2, 4, 5, 6, 7] might become [4, 5, 6, 7, 0, 1, 2]). Find the minimum element. You may assume that the array **has at least one element** and that **no duplicate exists** in the array. Your algorithm must run in **O(logN)**, where N is the number of elements in the array. For example:

Input: [3, 4, 5, 1, 2] 🡺 Output: 1

Input: [4, 5, 6, 7, 0, 1, 2] 🡺 Output: 0

Fill in the function “int MinimumInRotatedSortedArray(vector<int> &A)” in **Q5.cpp**.

**NOTE:** Upload the following files to DYS:

1. This file (Final.doc) [after filling in your name, ID and section]
2. Q1.cpp & Q1.h
3. Q2.cpp
4. Q3.cpp
5. Q4.cpp
6. Q5.cpp