AVL Tree

Given a set of keys, create a AVL tree with following requirements

1. For any node, one of the following condition is always satisfied

- both right and left subtree contain either equal number of keys
- number of keys in the right subtree are greater than number of keys in the left subtree by one

2. Identify a correct sequence of the insertions such that no rotations are required to maintain the AVL tree property

Consider the following example input

50

20

17

40

30

100

900

700

200

400

One of the correct sequences of insertion is as follows. Note that there could be multiple correct sequences. Finding out one such sequence is sufficient. Your code should take less than 30 seconds to run. In other words, do not use a brute force solution!

50

400

20

100

700

30

17

40

200

900

How your code should compile: g++ main.cpp

How your code should run: ./a.out <input1.txt >output1.txt

You can check that the binary search tree created after each insertion is also a valid AVL tree.

Expected output:

```
Sequence of insertions: 50, 400, 20, 100, 700, 30, 17, 40, 200, 900
Tree structure
Node: 50   Parent: NULL
Left Child: 20  Nodes in Left subtree: 4   Right Child: 400  Nodes in Right subtree: 5

Node 20 Parent: 50
Left Child: 17  Nodes in Left subtree: 1   Right Child: 30  Nodes in Right subtree: 2

Node: 400 Parent: 50
Left Child: 100  Nodes in Left subtree: 2   Right Child: 700  Nodes in Right subtree: 2

Node: 17 Parent: 20
Left Child: NULL Nodes in Left subtree: 0   Right Child: NULL Nodes in Right subtree: 0

Node: 30 Parent: 20
Left Child: NULL Nodes in Left subtree: 0   Right Child: 40 Nodes in Right subtree: 1

Node: 100 Parent: 400
Left Child: NULL Nodes in Left subtree: 0   Right Child: 200 Nodes in Right subtree: 1

Node: 700 Parent: 400
Left Child: NULL Nodes in Left subtree: 0   Right Child: 900 Nodes in Right subtree: 1

Node: 40 Parent: 30
Left Child: NULL Nodes in Left subtree: 0   Right Child: NULL Nodes in Right subtree: 0

Node: 200 Parent: 100
Left Child: NULL Nodes in Left subtree: 0   Right Child: NULL Nodes in Right subtree: 0

Node: 900 Parent: 700
Left Child: NULL Nodes in Left subtree: 0   Right Child: NULL Nodes in Right subtree: 0
```