Assignment 2 (COMP SCI 2201 and COMP SCI 7201): C++ Implementation of AVL Trees (worth 10%)

Due: 23:59PM 6th May 2018, late submissions will have penalties

1 Task Description

You are asked to use C++ to implement

- Binary Tree Traversal
- AVL Tree Insertion and Deletion

2 Submission Guideline

This is an individual assignment.
You must follow this guideline! Your submission will be marked automatically. Failure to follow this guideline will result in 0.

Your submission should contain a `main.cpp` file.

Your need to follow the binary tree and AVL tree concepts described to you in the lecture slides.You do not need to submit a design.

You should start your program by initializing an empty AVL tree. Your program takes one line as input as a command line argument. The input line contains n "modification moves" separated by spaces ($1 \leq n \leq 100$). The available modification moves are

- `Aint` (Character `A` followed by an int value between 1 and 100): `A3` means insert value 3 into the AVL tree. If 3 is already in the tree, do nothing.
- `Dint` (Character `D` followed by an int value between 1 and 100): `D3` means delete value 3 from the AVL tree. If 3 is not in the tree, do nothing.

Your input is then followed by exactly one finishing move (`PRE` or `POST` or `IN`): If the finishing move is `PRE`, then you should print out the tree (in its current situation) in pre-order. If the tree is empty, print out `EMPTY`. Otherwise, print out the values separated by spaces. `POST` and `IN` are handled similarly.

You don't need to worry about invalid inputs. We wil be executing your code as follows:

Sample input 1: `A1 A2 A3 IN`
Sample output 1: `1 2 3`
Sample input 2: `A1 A2 A3 PRE`
Sample output 2: `2 1 3`
Sample input 3: `A1 D1 POST`
Sample output 3: `EMPTY`

3 Marking

Marking will be done automatically. The total mark is 10.

- 9 marks for code correctness: Your code will be tested against a random set of test cases.
- 1 mark for compilation

4 SVN Instructions

First of all, you need to create a directory under version control:

```
svn mkdir --parents -m "Creating ADSA Assignment 2 folder" https://version-control.adelaide.edu.au/svn/aXXXXXXX/2018/s1/adsa/assignment2/
```

aXXXXXXX should be your student ID. The directory path needs to be exactly "2018/s1/adsa/assignmentK", where "K" is the assignment number. To check out a working copy, type

```
svn checkout https://version-control.adelaide.edu.au/svn/aXXXXXXX/2018/s1/adsa/assignment2/ adsa-18-s1-assignment2/
cd adsa-18-s1-assignment2
svn add *.cpp
```

Commit the files to SVN:

```
svn commit -m "Adding ADSA assignment 2"
```

SVN helps keeping track of file changes (over different commits). You should commit your work early and often.

5 Web submission

You are asked to submit via the web interface `https://cs.adelaide.edu.au/services/websubmission/`. The submission steps should be self-explanatory. Simply choose the correct semester, course, and assignment. The websubmission system will automatically fetch the latest version of your work from your SVN repository (you may also choose to submit older versions). Once your work is submitted, the system will launch a script checking the format of your submission. Click "View Feedback" to view the results. You are welcome to resubmit for as many times as you wish (before the deadline).

We will compile your code using `g++ -o main -std=c++11 -O2 -Wall *.cpp`. It is your responsibility to ensure that your code compiles **on the university system**.[1]

[1] g++ has too many versions, so being able to compile on your laptop does not guarantee that it compiles on the university system. You are encouraged to debug your code on a lab computer (or use SSH).