C++ Implementation of AVL Trees (worth 10%, due Oct 2nd 23:59PM, late submissions not accepted)

Mingyu Guo

1 Task Description

You are asked to use C++ to implement

- Binary Tree Traversal
- AVL Tree Insertion and Deletion

2 Submission Guideline

You must follow this guideline! Your submission will be marked automatically. Failure to follow this guideline will result in 0.

Your submission should contain exactly two files: `main.cpp` and `test.txt`.

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. 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 into 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.

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`

You are responsible for writing your own test cases. Please submit `test.txt`. This file should contain up to 100 lines. Each line must start with a sequence of modification moves, finished by a finishing move, and then followed by the correct output.

For example, the following three lines represent the above three samples.

```
A1 A2 A3 IN 1 2 3
A1 A2 A3 PRE 2 1 3
A1 D! POST EMPTY
```

3 Marking

Marking will be done automatically. The total mark is 5.

- Students will be randomly divided into ten groups.
- 3 marks for code correctness: We will collect all test cases submitted by students from your group. Your code will be tested against them. If your code passes all test cases, then you receive 3 marks. If your code passes 90% test cases, then you receive 2 marks. If your code passes 80% test cases, then you receive 1 mark.
- 2 marks for test case quality: If your test cases contain mistakes, then your receive 0 marks. We will run your test cases against other students' submissions (only those from the same group). We keep track of how many submissions you test cases can break (by "breaking" a submission, I mean at least one of your test cases proves that the submission is buggy). Let x be the number of buggy submissions (our definition of buggy: a submission is not buggy if and only if passes all test cases). If your test cases can catch all buggy submissions, then you receive 2 marks. If your test cases can catch 80% buggy submissions, then you receive 1 mark.

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/2017/s2/adsa/assignment2/
```

aXXXXXXX should be your student ID. The directory path needs to be exactly "2017/s2/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/2017/s2/adsa/assignment2/ adsa-17-s2-assignment2/
cd adsa-17-s2-assignment2
svn add main.cpp
svn add test.txt
```

Commit the files to SVN:

```
svn commit -m "Adding ADSA assignment 2 main.cpp test.txt"
```

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

5 Websubmission

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. Your mark will be calculated offline after the deadline. You are welcome to resubmit for as many times as you wish (before the deadline).

We will compile your code using `g++ -o main.out -std=c++11 -O2 -Wall main.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).