**A Letter to the students of**

**Software Engineering Practice**

Dear Students,

Welcome to NCU and thanks for enrolling in this class. Since there are students from different countries, schools, and cultures, we would like to lay down the rules of plagiarism in this class.

**So, what you CAN do?**

1. You can discuss the lab, problems, possible solutions and to your classmates but you cannot read or share your code(or report) to your classmates.
2. You should complete the code(or report) alone.

In theory, by following the above rules, it is impossible to have your code(or report) sharing the same/similar portions of materials.

**Standards of Plagiarism:**

There are no standards. Determining plagiarism is one thing that how much efforts we want to spend on catching such dishonest activities. So, if your intension is not learning but credits, we suggest you to withdraw this class.

**Past Real Cases**

Several students submit the same report/code and claim that they worked together to come out the results. Sorry, we do not accept such excuses. All the students will get zero points in the homework.

So, if you are really excellent and always produce an original piece of work before everyone and you also have a warm and kind heart to help classmates, you can tell them how to solve the problems verbally. You can even help them debug their programs. Only by this way, your original piece of work will never leak out. **Protect yourself and your right**.

**Debugging Lab**Debug an AVL implementation

**Description:**

In this lab, a program’s source code called ***AVLtree-incorrect.java*** is provided for you. The code contains an implementation of an AVL tree. AVL tree is a very basic data structure. If this program can be executed correctly, you can interact with it as in Table 1. Unfortunately, this program is wrong. There are two bugs hidden in the program.

**Your Tasks:**

1. If you totally forget what is an AVL tree, please google “AVL tree” to recall its concept and idea.
2. Please run the AVLtree-incorrect.java successfully in your computer first. If you are capable of finding and fixing the bugs immediately, please do so. No one stops you. However, typically you need to come up with some test runs to reproduce the bugs first. These test runs can produce some incorrect outputs. However, using your brain you should be able to derive manually the **expected correct outputs** for the test runs.
3. You should use debugger and set break points to debug. Of course, this is comparatively a very small program. No body prevents you from finding the bug by code reading. To be honest with you, I hide the bugs very well. It is not easy to find the bugs immediately if you insist to do so.
4. Once you fix the bugs, please follow TA’s instructions to submit your files.
5. TA will grade your work based on:
   1. Whether you fix the bugs successfully

**HINTS:**

1. If your tests are not complete, you may not catch all the bugs.
2. Table 1 actually gives you a lot of test scenarios which are suitable for validating the correctness of your fixes.

Table 1. The correct behaviors of AVL tree. (**bold font** means input given by human, no bold-font words are program output.

|  |
| --- |
| AVLTree Tree Test      AVLTree Operations    1. insert  2. search  3. count nodes  4. check empty  5. clear tree  **4**  Empty status = true    Post order :  Pre order :  In order :  Do you want to continue (Type y or n)    **y**    AVLTree Operations    1. insert  2. search  3. count nodes  4. check empty  5. clear tree  **1**  Enter integer element to insert  **10**    Post order : 10  Pre order : 10  In order : 10  Do you want to continue (Type y or n)    **y**    AVLTree Operations    1. insert  2. search  3. count nodes  4. check empty  5. clear tree  **1**  Enter integer element to insert  **9**    Post order : 9 10  Pre order : 10 9  In order : 9 10  Do you want to continue (Type y or n)    **y**    AVLTree Operations    1. insert  2. search  3. count nodes  4. check empty  5. clear tree  **1**  Enter integer element to insert  **8**    Post order : 8 10 9  Pre order : 9 8 10  In order : 8 9 10  Do you want to continue (Type y or n)    **y**    AVLTree Operations    1. insert  2. search  3. count nodes  4. check empty  5. clear tree  **1**  Enter integer element to insert  **7**    Post order : 7 8 10 9  Pre order : 9 8 7 10  In order : 7 8 9 10  Do you want to continue (Type y or n)    **y**    AVLTree Operations    1. insert  2. search  3. count nodes  4. check empty  5. clear tree  **1**  Enter integer element to insert  **6**    Post order : 6 8 7 10 9  Pre order : 9 7 6 8 10  In order : 6 7 8 9 10  Do you want to continue (Type y or n)    **y**    AVLTree Operations    1. insert  2. search  3. count nodes  4. check empty  5. clear tree  **1**  Enter integer element to insert  **5**    Post order : 5 6 8 10 9 7  Pre order : 7 6 5 9 8 10  In order : 5 6 7 8 9 10  Do you want to continue (Type y or n)    **y**    AVLTree Operations    1. insert  2. search  3. count nodes  4. check empty  5. clear tree  **1**  Enter integer element to insert  **4**    Post order : 4 6 5 8 10 9 7  Pre order : 7 5 4 6 9 8 10  In order : 4 5 6 7 8 9 10  Do you want to continue (Type y or n)    **y**    AVLTree Operations    1. insert  2. search  3. count nodes  4. check empty  5. clear tree  **1**  Enter integer element to insert  **3**    Post order : 3 4 6 5 8 10 9 7  Pre order : 7 5 4 3 6 9 8 10  In order : 3 4 5 6 7 8 9 10  Do you want to continue (Type y or n)    **y**    AVLTree Operations    1. insert  2. search  3. count nodes  4. check empty  5. clear tree  **1**  Enter integer element to insert  **2**    Post order : 2 4 3 6 5 8 10 9 7  Pre order : 7 5 3 2 4 6 9 8 10  In order : 2 3 4 5 6 7 8 9 10  Do you want to continue (Type y or n)    **y**    AVLTree Operations    1. insert  2. search  3. count nodes  4. check empty  5. clear tree  **1**  Enter integer element to insert  **1**    Post order : 1 2 4 6 5 3 8 10 9 7  Pre order : 7 3 2 1 5 4 6 9 8 10  In order : 1 2 3 4 5 6 7 8 9 10  Do you want to continue (Type y or n)    **y**    AVLTree Operations    1. insert  2. search  3. count nodes  4. check empty  5. clear tree  **1**  Enter integer element to insert  **0**    Post order : 0 2 1 4 6 5 3 8 10 9 7  Pre order : 7 3 1 0 2 5 4 6 9 8 10  In order : 0 1 2 3 4 5 6 7 8 9 10  Do you want to continue (Type y or n)    y    AVLTree Operations    1. insert  2. search  3. count nodes  4. check empty  5. clear tree  3  Nodes = 11    Post order : 0 2 1 4 6 5 3 8 10 9 7  Pre order : 7 3 1 0 2 5 4 6 9 8 10  In order : 0 1 2 3 4 5 6 7 8 9 10  Do you want to continue (Type y or n)    y    AVLTree Operations    1. insert  2. search  3. count nodes  4. check empty  5. clear tree  2  Enter integer element to search  12  Search result : false    Post order : 0 2 1 4 6 5 3 8 10 9 7  Pre order : 7 3 1 0 2 5 4 6 9 8 10  In order : 0 1 2 3 4 5 6 7 8 9 10  Do you want to continue (Type y or n)    y    AVLTree Operations    1. insert  2. search  3. count nodes  4. check empty  5. clear tree  2  Enter integer element to search  4  Search result : true    Post order : 0 2 1 4 6 5 3 8 10 9 7  Pre order : 7 3 1 0 2 5 4 6 9 8 10  In order : 0 1 2 3 4 5 6 7 8 9 10  Do you want to continue (Type y or n)    y    AVLTree Operations    1. insert  2. search  3. count nodes  4. check empty  5. clear tree  5    Tree Cleared    Post order :  Pre order :  In order :  Do you want to continue (Type y or n)    y    AVLTree Operations    1. insert  2. search  3. count nodes  4. check empty  5. clear tree  4  Empty status = true    Post order :  Pre order :  In order :  Do you want to continue (Type y or n)    n |