Due : Oct. 29 (10/29), 13:00
Late submission due: Oct. 30 (10/30), 13:00

Overview

This assignment consists of 2 parts.

General Notes

- *Read this homework guideline carefully.* If you do not follow the guidelines, you may receive a 0 regardless of whether your code works or not.
- Do not use any IDEs (Eclipse, IntelliJ IDEA, etc.)
  - We recommend Sublime Text (Linux/Mac/Windows), Atom (Linux /Mac/Windows), Notepad++ (Windows), or TextWrangler (Mac).
  - IDEs often create a "package" of your code, which breaks the auto-grader.
  - **If you know how to fix the package problem**, you can use any IDE you want. However, we will not answer any questions related to this problem since we have already recommended a solution.
- Do not change any method or class signatures. If your code changes any class or method names or signatures, you will receive an automatic 0.
- Make sure your code compiles. Non-compiling code will automatically receive a 0. If you have a problem that is causing you to not be able to compile, it may be better to just comment out the incorrect code and return a dummy value (something like null or -1) so the rest can compile.
- To ensure that your code will be accepted by the autograder, you should submit your code on LearnUS, download it again, recompile it and check the provided test suite. This way, you know that the file you are submitting is the correct one.
- You can use any course materials. However, if you do not cite the source you referred to, it might be checked as copied code. Please write the material (and page) you referred in comments.

1 Double-ended Queues

A double-ended queue (deque) is a queue-like data structure the supports insertion and deletion at both the front and the back of the queue (from the textbook).

You will implement a deque that stores an integer (which will be given at the construction of deque) of data in Deque.java file. You must implement the following methods as well as its constructor:

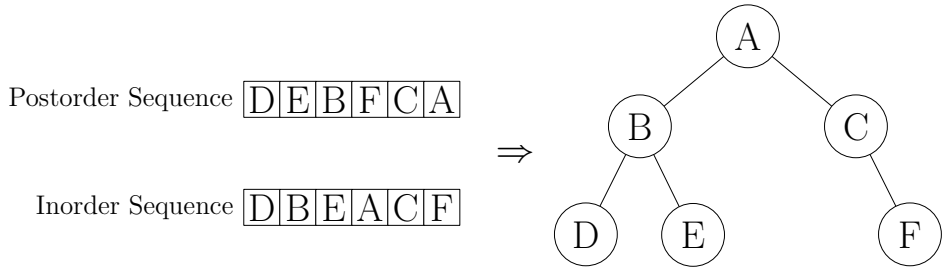
- `insertFirst`: Insert the given item at the front of the deque.
- `insertLast`: Insert the given item at the back of the deque.
- `deleteFirst`: Delete the item at the front of the deque. If the deque is empty, raise an exception.
- `deleteLast`: Insert the given item at the back of the deque. If the deque is empty, raise an exception.
- `first`: Return the item at the front of the deque. If the deque is empty, raise an exception.
- `last`: Return the item at the back of the deque. If the deque is empty, raise an exception.
- `size`: Return the number of items in this deque.

Note that every method must run in $O(1)$ time (regardless to the number to the number of items the deque has). You will be provided with a Node class with the following methods. You must store the given values only in the nodes.

- `setNext`: Set the given node as the next node.
- `setPrev`: Set the given node as the previous node.
- `setVal`: Set the given item as the value of this node.
- `next`: Return the next node.
- `prev`: Return the previous node.
- `val`: Return the value of this node.

2 Reconstruction

Given (1) a length n postorder traversal sequence and (2) a length n inorder traversal sequence of a binary tree, you can reconstruct the original binary tree of n nodes.



Given the two sequences given in array, you must implement the following method in provided Reconstructor.java file:

- `reconstruct`: Reconstruct the tree from the given two sequences. If there is no such tree, raise an exception.

The sequence will be given as a int array that has an unique id that indicates each node in the tree. You will be provided with a TreeNode class with the following methods. You must store the given id in each of the nodes.

- `setLeft`: Set the given node as the left child node.
- `setRight`: Set the given node as the right child node.
- `setParent`: Set the given node as the parent node.
- `setVal`: Set the given item as the value of this node.
- `left`: Return the left child node.
- `right`: Return the right child node.
- `parent`: Return the parent node.
- `val`: Return the value of this node.

General Directions

- Write your name and student ID number in the comment at the top of the files you submit.
- Implement all of the required methods.
- You should not import anything that is not already included in the file.
- Pay careful attention to the required return types and edge cases.
- All the codes we provide can be found in src/base directory. If you are unsure what a class/method exactly does, please refer to the code.
- You are free to implement any algorithm that you wish, but you must code it yourself. If you referred to any course materials, you must write the name of the material and page in comments. We will only be testing that your code produces a correct result and terminates in a reasonable amount of time.

Submission Procedure

You *must* make a zip file for submission using Gradle build tool (refer to Compiling section). For this assignment, the zip file will contain only the following three files:

- Deque.java
- Reconstructor.java
- your_student_id_number.txt

You must rename 2021xxxxxx.txt to your actual student ID number. Inside of that text file, you must include the following text. Please be sure to write all the following text including the last period.

In completing this assignment, I pledge that I have not given nor received any unauthorized assistance.

If this file is missing, you will get 0 on the assignment. It should be named *exactly* your student id, with no other text. For example, *2021123456.txt* is correct while something like *2021123456_pa2.txt* will receive 0.

Compiling

This assignment uses Gradle build tool to automate compiling and testing procedure. The following command will test your Java code against the provided test suite:

```
% ./gradlew -q runTestRunner
```

The following command will zip the files for your submission. The zip file will be named with your student id (the name of .txt file) and will lie in "build" directory. Be aware that the command will be interrupted if your pledge does not comply the guideline.

```
% ./gradlew -q zipSubmission
```

Since the testrunner blocks the standard output from printing, it is hard to test your code fragment while writing the code. For this purpose, we also provide an empty Main class. As this file is not for submission, you may use any features Java provides. The following command will run the Main class instead of the test suite.

```
% ./gradlew -q --console=plain runMain
```

On Windows, try `gradlew.bat` instead of `./gradlew` if you met an error. Moreover, you may omit the '`-q`' option to review the compile log.

Testing

We have provided a small test suite (src/test) for you to check your code. You can test your code by the means mentioned above.

Note that the test suite we will use to grade your code is much more rigorous than the one provided here (and not necessarily a superset of the provided tests). You should consider making your own test suites to check your code more thoroughly.