Due : Sept. 26 (9/26), 13:00
Late submission due: Sept. 27 (9/27), 13:00

Overview

This assignment consists of two parts, making a Round-Robin scheduler and a Knight's tour solver.

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 YSCEC, 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 However, if you do not cite the source you referred to, it might be checked as copied code.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 Round-Robin Scheduler

Scheduling is important to use your time efficiently. Computers also use scheduling to use its resources efficiently. The Round-Robin scheduler is one of the schedulers that a computer uses.

First, you will implement a Circular Doubly Linked List and second, use it for your simulation in the scheduler.

1.1 Circular Doubly Linked List

A circular doubly linked list is circle-shaped list whose node has two pointers pointing its next and previous nodes.

You will implement this type of list that stores integers in CDLList.java file using the Node class that has two pointer variables `prev`, `next` and an integer variable `value`.

You must implement the following methods as well as its constructor:

`insert`: Insert the given integer between its head position and the position before head (the head must not be changed). If there is no head, then use it as head.

`delete`: Delete the previous node of the head node. If the list is empty, raise an exception.

`getHead`: Return the head node. If there is no head, raise an exception.

`rotateForward`: Make the head point its next. If there is no head, raise an exception.

`rotateBackward`: Make the head point its previous. If there is no head, raise an exception.

`size`: Return the number of items in the list.

Note that your list must contain exactly the items the list must have. For example, you cannot use any dummy node in your implementation. Also, you must return the head reference from `getHead` method. If you create a copy of internal representation for each call, you may fail the test.

1.2 Round-Robin Scheduler

Round-Robin scheduler works by assigning each job with equal portions of time slices and the job is processed in circular order with no priority. Here you will simulate a Round-Robin Scheduler, with some additional operations.

Now, using the above data structure, you have to simulate a Round-Robin Scheduler in the RRScheduler.java file. We will assume that each job is assigned to one time segment. In other words, if one time segment passes, then one job is processed and moves on to the next job. This scheduler has the following operations to simulate:

`insert`: Insert the given job id at the end of the scheduler.

`done`: One time segment passes and the job processed is deleted.

`timeflow`: Given an integer n, simulate the time segment that many times. In other words, n time segment passes.

`changeDirection`: Change the direction of the scheduler. If it was going in order, change it to reverse. If it was going in reverse, change it back to normal.

`currentJob`: Return the current job id. If there is no current job, raise an exception.

2 Knight's tour

The knight piece in the game of chess has a unique move compared to other chess pieces. It may move two squares in vertically(horizontally) and one square horizontally(vertically).

The Knight's tour is a problem of finding the path of the knight on a given board so that every squares of the board is visited exactly once. In other words, you have to find the order of the squares visited by the knight.

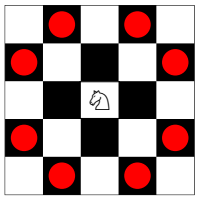

Figure 1: Positions a knight can jump to.

Here you will solve a Knight's tour problem by recursion in the TourSolver.java file. You must implement the `solve` method that solves the Knight's tour problem on the given board which might have some missing squares. You should return a sequence of the visited squares id. If you cannot solve the problem on the given board, you should return an empty sequence, `not null`.

We provided a class for $N \times M$ board with the following methods:

`getWidth`: Return the width of the board.

`getHeight`: Return the height of the board.

`isMissing`: Receive (int x, int y) and return if the given square coordinate (starting from (0,0)) exists. If the given square is missing, then the method returns `true` and if not, returns `false`. If you provide a wrong coordinate, it raises an `IndexOutOfBoundsException`.

`squareId`: Receive (int x, int y) and returns the id of the given square coordinate (starting from (0,0)). If you provide a wrong coordinate, it raises an `IndexOutOfBoundsException`.

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 four files:

- CDLList.java
- RRScheduler.java
- TourSolver.java
- your_student_id_number.txt

You must rename 2020xxxxxx.txt to your actual student ID number. Inside of that text file, you must include the following text and write your name at the bottom. 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, *2020123456.txt* is correct while something like *2020123456_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 testcases:

```
% ./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 testcases.

```
% ./gradlew -q 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 small testcases (src/test) for you to check your code. You can test your code by the means mentioned above.

Note that the testcases 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 cases to check your code more thoroughly.