Due : Nov. 19 (11/19), 13:00
Late submission due: Nov. 20 (11/20), 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 However, if you do not cite the source you referred to, it might be checked as copied code.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 Heap

A heap is a complete tree that satisfies the following property: for any given node N of the tree and its parent node P, the key of P is less than or equal to the key of N. Also, it is an efficient implementation of an abstract data type called a priority queue.

1.1 Heap

You will implement a min heap that stores entries whose key is an integer in **Heap.java** file. You must implement the following methods as well as its constructor:

- `bottomUp(keys, values)`:
  Construct a heap with entries $(keys[i], values[i])$ pairs using bottom-up construction. If the heap is not empty or lengths of $keys$ and $values$ are different, throw an exception.
- `insert(k, v)`: Insert an entry with key k and value v to the heap. Each entry in the heap will be given a unique key.
- `removeMin()`: Remove the entry with the smallest key in the heap and return the value of the removed entry. If the heap is empty, throw an exception.
- `min()`: Return the value of the entry with the smallest key in the heap. If the heap is empty, throw an exception.
- `size()`: Return the number of entries in the heap.
- `isEmpty()`: Return whether or not the heap is empty.
- `getRoot()`: Return the root node of the heap. If the heap is empty, throw an exception.

Note that the running time of the construction must be in $O(n)$ for the number n of given entries, and the `insert` and `removeMin` methods must be in $O(\log m)$ for the size m of the heap. You will be provided with a Node class with the following methods (5 setters and getters each):

- `setParent(n)/getParent()`: Set n as the parent. / Get the parent node.
- `setLeft(n)/getLeft()`: Set n as the left child. / Get the left child.
- `setRight(n)/getRight()`: Set n as the right child. / Get the right child.
- `setKey(k)/getKey()`: Set k as the key. / Get the key.
- `setValue(v)/getValue()`: Set v as the value. / Get the value.

2 Hash Table

In computing, a hash table (hash map) is a data structure that implements an associative array abstract data type, a structure that can map keys to values. A hash table uses a hash function to compute an index, also called a hash code, into an array of buckets or slots, from which the desired value can be found.

2.1 Hashing

In this assignment, you will build a hash table of an arbitrary type (given at the construction) with an hash function and resize function in Hash.java file. For simplicity, we will only store the key in the hash table.

For the hashing of the key, use the hash method interface in the **IHash-Function.java** file. If a collision occurs, use linear probing to avoid the collision. Also, you must **resize** the hash table if the table has too many items. Do this by using the resize method interface in the **IResizeFunction.java** file. When the resize happens, you must **rehash** the table. The rehashing order starts from index 0 to the last index of the table at the time of the rehash.

The hash function and resize function will be given on construction. If they are not given, you must make your own hash function and resize function for the hash table. Implement the following methods as well as its constructor in the Hash.java file. Each method should run in $O(1)$ except the show method:

- `put(k)`: Insert the key k into the hash table. If a collision occurs, use linear probing to avoid collision.
- `remove(k)`: Find the key k and delete the key from the hash table. If the key does not exist, raise an exception.
- `exists(k)`: If the key k exists in the table, return `true`. Otherwise, return `false`.
- `size()`: Return the number of keys in the hash table.
- `tablesize()`: Return the size of the table.
- `show()`: Return the keys in the hash table as an array. The index of each key in the list must be the same index of the bucket index of the key in the hash table. If a bucket has no item, assign `null` on that list index.

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:

- Heap.java
- Hash.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_pa3.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.