Comp 251: Assignment 1

Answers must be returned online by October 9th (11:55pm), 2018.

General instructions (Read carefully!)

- Your solution must be submitted electronically on MyCourses.
- You are provided some starter code that you should fill in as requested. Add your code only where you are instructed to do so. You can add some helper methods. Do not modify the code in any other way and in particular, do not change the methods or constructors that are already given to you, do not import extra code and do not touch the method headers. The format that you see on the provided code is the only format accepted for programming questions. **Any failure to comply with these rules will give you an automatic 0.**
- The starter code includes a tester class. If your code fails those tests, it means that there is a mistake somewhere. Even if your code passes those tests, it may still contain some errors. We will grade your code with a more challenging set of examples. We therefore highly encourage you to modify that tester class, expand it and share it with other students on the myCourses discussion board. Do not include it in your submission.
- Your code should be properly commented and indented.
- **Do not change or alter the name of one of the files you must submit**. Files with the wrong name will not be graded. Make sure you are not changing file names by duplicating them. For example, main (2).java will not be graded. Make sure to double-check your zip file.
- Do not submit individual files. Include all your files into a .zip file and, when appropriate, answer the complementary quiz online on MyCourses.
- **You will automatically get 0 if the files you submitted on MyCourses do not compile.**
- To some extent, collaborations are allowed. These collaborations should not go as far as sharing code or giving away the answer. You must indicate on your assignments (i.e. as a comment at the beginning of your java source file) the names of the people with whom you collaborated or discussed your assignments (including members of the course staff). If you did not collaborate with anyone, you write "No collaborators". If asked, you should be able to orally explain your solution to a member of the course staff.
- This assignment is due on October 9^{th} at 11h55pm. It is your responsibility to guarantee that your assignment is submitted on time. We do not cover technical issues or unexpected difficulties you may encounter. Last minute submissions are at your own risk.
- Multiple submissions are allowed before the deadline. We will only grade the last submitted zip file. Therefore, we encourage you to submit as early as possible a preliminary version of your solution to avoid any last minute issue.
- Late submissions will receive a penalty of 20% per day. We will not accept any submission more than 72 hours after the deadline. The submission site will be closed, and there will be no exceptions, except medical.

- In exceptional circumstances, we can grant a small extension of the deadline (e.g. 24h) for medical reasons only. However, such request must be submitted before the deadline, and justified by a medical note from a doctor, which must also be submitted to the McGill administration.
- Violation of any of the rules above may result in penalties or even absence of grading. If anything is unclear, it is up to you to clarify it by asking either directly the course staff during office hours, by email at (`cs251@cs.mcgill.ca`) or on the discussion board on myCourses (recommended). Please, note that we reserve the right to make specific/targeted announcements affecting/extending these rules in class and/or on the website. It is your responsibility to monitor the course website and MyCourses for announcements.
- The course staff will answer questions about the assignment during office hours or in the online forum on MyCourses. We urge you to ask your questions as early as possible. We cannot guarantee that questions asked less than 24h before the submission deadline will be answered in time. In particular, we will not answer individual emails about the assignment that are sent sent the day of the deadline.

Questions

Exercise 1 (80 points) We want to compare the performance of hash tables implemented using chaining and open addressing. In this assignment, we will consider hash tables implemented using the multiplication and linear probing methods. We will (respectively) call the hash functions h and g and describe them below. Note that we are using the hash function h to define g.

$$\begin{aligned} \text{Collisions solved by chaining (multiplication method):} \quad & h(k) = ((A \cdot k) \mod 2^w) >> (w - r) \\ \text{Open addressing (linear probing):} \quad & g(k, i) = (h(k) + i) \mod 2^r \end{aligned}$$

In the formula above, r and w are two integers such that $w > r$, and A is a random number such that $2^{w-1} < A < 2^w$. In addition, let n be the number of keys inserted, and m the number of slots in the hash tables. Here, we set $m = 2^r$ and $r = \lceil w/2 \rceil$. The *load factor* α is equal to $\frac{n}{m}$.

We want to estimate the number of collisions in random sequences of insertions and deletions of keys with respect to the choice of values for w and α.

We provide you a set of three template files within `COMP251HW1.zip` that you will complete. This file contains three classes, a main class and one for each hash function. Those contain several helper functions, namely `generateRandom` that enables you to generate a random number within a specified range. Details on which functions are included, how to use them, and where to add in your code can be found as comments in the java files. Please read them with attention. In addition, we provide you a jar file to visualize your results named `JavaPlotBuilder.jar`.

Your first task is to complete the two java methods `Open_Addressing.probe` and `Chaining.chain`. These methods must implement the hash functions for (respectively) the linear probing and multiplication methods. They take as input a key k, as well as an integer $0 \leq i < m$ for the linear probing method, and return a hash value in $[0, m[$. Note that the value of A must be updated when you change w.

Next, you will implement the method `insertKey` in both classes, which inserts a key k into the hash table and returns the number of collisions encountered before insertion. Note that for this exercise

as well as for the rest of the homework, we define the number of collisions as the number of keys encountered, or "jumped over" before inserting or removing a key. You can assume the key is not negative.

You will also implement a method `removeKey`, this one only in `Open_Addressing`. This method should take as input a key k, and remove it from the hash table while visiting the minimum number of slots possible. Like `insertKey`, it should output the number of collisions. If the key is not in the hash table, the method should simply not change the hash table, and output the number of slots visited. You will notice from the code and comments that empty slots are given a value of -1. If applicable, you are allowed to use a different notation of your choice for slots containing a deleted element.

Finally, you will complete the method `main.main`, which calls the previous functions from the main class. There are three tasks to complete within this main method.

Task 1

First, you will test the effect of increasing the number of keys on the average number of collisions for each hash function. You will be given an array of keys to insert, `keysToInsert`, and a list of values of n to test, `nList` inside the main method. Random seeds can be used in java in order to make random results reproducible (and eventually evaluate assignments). You will find the hash tables already initialized with such seed. For each value of `n`, insert the `n` first elements of `keysToInsert` into each hash table, and store the α value, as well as the average number of collisions for that value of α into the appropriate provided list. The program is already set up to output a CSV file for you to visualize.

Task 2

Your second task is to test the `removeKey` method on the `Open_Addressing` table from task 1 with n=16. Initialize a new `Open_Addressing` hash table with the same seed as in Task 1, and insert the first 16 elements of `keysToInsert`. You will be given an array of keys to remove named `keysToRemove`. Call `removeKey` with each of these keys. Store the number of collisions associated with each removal operation in the arraylist removeCollisions, as well as the index of the key you just attempted to remove in removeIndex. Finally, use the provided method to output a CSV file.

Task 3

Your third task is to evaluate the effect of varying w on the number of collisions for each method. For this exercise, the keys you insert will be generated randomly with `generateRandom`. Each key can be inserted only once (i.e. The random sequence of keys must have no duplicates). For this exercise, you will not be using a specific seed, which you can do by calling functions that require a seed argument with a seed of -1. Because your experiments will now have some variance, you will need to execute 10 simulations for each value of w to obtain representative averages. You will choose appropriate values of w, use the provided function to output a CSV file, and visualize your results. You will submit a pdf file, called `Conclusions.pdf`, including plots of your results, as well as a short explanation of what you observe, and an explanation for it.

To plot your results, you can use `JavaPlotBuilder.jar` to generate a plot with α on the x-axis and the average number of collision on the y-axis. `JavaPlotBuilder.jar` is run from command line as follows:

```
java -jar JavaPlotBuilder.jar filename.csv
```

where filename.csv is the name of the file that is generated by the main method.

For this assignment, you will need to submit a zip file containing the completed version of the three provided java files, `n_comparison.csv`, `remove_collisions.csv`, and `w_comparison.csv` (the three CSV files generated by the main method), as well as `Conclusions.pdf`, the PDF file with your observations and explanations.

Once you have submitted your files, you can proceed to the second part of the assignment.

Exercise 2 (20 points) *This section is answerable through MyCourses. Note that you MUST use your own results to answer those questions. Answers to this quiz that would not match the results presented in your pdf file will be considered plagiarism (refer to course outline).*