CISC-235
Assignment 3

Due Date: March 19, 2017

In this assignment you will experiment with hash functions.

You have been hired as a consultant for a new government agency called the Canadian Intelligence and Secrecy Council (CISC). Your assigned task is to design a data structure for storing information about their agents. Each agent is identified by a seven-letter or eight-letter English word codename (such as "curling" or "sweepers"). You have not been told what the data associated with each codename will be – your security clearance is not high enough.

There are 2500 codenames to be stored. A sample set of agent codenames is provided in the file top_secret_agent_codenames_2017.txt. There is **no way** this is the actual set of agent codenames, no, not at all.

The structure you choose must allow insertion and searching, but for the purpose of this assignment we will focus on insertions. **To satisfy performance requirements, each insertion operation must look at no more than 10 possible locations for the new value**. Upon reviewing your CISC-235 notes you have realized that a binary search tree cannot satisfy this requirement, so you have settled on a hash table.

The Director of CISC was previously a Computer Science professor and she has taken an interest in your project. She has already decided that you are required to use some form of open addressing.

She wants you to explore at least two forms of open addressing: quadratic probing and double hashing. For each method she wants you to determine a table size that lets you achieve the required performance standard. See below for a discussion of how to compute the necessary information.

The Director has also set you an interesting challenge: if you can achieve the required performance level with a table size no greater than 3000, she will double your consulting fee. (Note: this has no bearing on your grade for this assignment.)

Part 1:

Decide how you will convert the codewords into integers. We discussed this problem in class – you will also find a wealth of ideas on the Internet. Whatever method you decide on, explain why you chose it, and remember to cite your source if it is not your own creation.

Part 2:

Implement a hash table where collisions are resolved by quadratic probing.

Use an h'(k) hashing function of your own choice. You must implement the algorithm yourself. Using downloaded code from external sources is not permitted – but writing your own code based on a published algorithm is fine (remember to cite your sources).

Try **at least** three combinations of c_1 and c_2 :

$c_1 = 1, c_2 = 1,$

$c_1 = 2, c_2 = \frac{1}{2}$ (remember that you will have to convert the result to an integer),

other combinations of your choice.

For each combination, try to **find a table size that lets you achieve the requirement on maximum probe sequence length. Use experimentation to get close to the minimum table size that satisfies the requirement.**

Part 3:

Repeat Part 2, but this time with Double Hashing. Try at least three combinations of h'(k) and h″(k). For each combination, find a table size that achieves the required performance.

You are free to choose any functions you like for h'(k) and h″(k), but as with Part 2 you must implement them yourself.

Part 4:

Discuss the results of your experiments. Do they support the hypothesis that Double Hashing allows us to use smaller tables than Quadratic Probing does, when trying to achieve a particular level of performance?

Computing the Average Number of Comparisons

Every time your program looks at the content of a table address, that counts as a comparison. So if you are inserting a value and you try addresses 17, 5, and 83 before finally inserting the value in address 36, that counts as **four** comparisons.

Since we don't know which keys are most likely to be searched for, we can assume that each key is equally likely to be the target of a search. This means that the average number of comparisons in a search operation will be exactly the same as the average number of comparisons in an insertion. To compute that, we can add up the number of comparisons made during all the insertions, and divide by the number of values that were inserted.

Logistics:

You may complete the programming part of this assignment in Python, Java, C or C++.

You must submit your source code, properly formatted and documented You must also submit a PDF file summarizing the results of your experiments and containing your conclusions. All files must contain your name and student number, and must contain the following statement: “I confirm that this submission is my own work and is consistent with the Queen's regulations on Academic Integrity.”

You are required to work individually on this assignment. You may discuss the problem in general terms with others and brainstorm ideas, but you may not share code. This includes letting others read your code or your written conclusions. The course TAs will be available during office hours to advise and assist you regarding this assignment.

The due date for this assignment is 11:59 PM, March 19, 2017.