CSUC CSCI-311 - Algorithms and Data Structures S24
Lab Assignment 6

Goal: 1) practicing using hashing by using chaining to deal with collisions. In particular, the hash table itself will be a vector of vectors of strings. We will use a simple polynomial rolling hash function.

Submission: C++ solutions to these problems should be written in a file called HashTable.cpp (a skeleton is available on Canvas along with HashTable.h and HashTableDriver.cpp) and submitted on inginious.

Note: submission on inginious is only used for grading, and you should do all the testing on the ecc-linux server. You should not use inginious for purpose of testing your code.

Coding Style: Note that your submission should use good C++ coding style and should include appropriate comments (for readability and maintainability). Specifically, your code must follow common C++ coding conventions for Naming, Indentation, and Comments. Points will be deducted if these are not present.

Collaboration: There will be time in lab to discuss these problems in small groups and I highly encourage you to collaborate with one another outside of class. However, you must write up your own solutions **independently** of one another. In addition, do not post solutions in any way. Also, please include a list of the people you work with in a comment section at the top of your submission.

Have fun!

Assignment Date: **Apr 1, 2025**
Due Date: **11:59pm on Apr 11, 2025**
Grace Due Date: **11:59pm on Apr 14, 2025**

Grading Notes: 1) Assignment is due on the Due Date. 2) For each day late after the Due Date, there will be 10% penalty on the assignment's grades. 3) Submission is not accepted after the Grace Date. In other words, you will receive zero pts if your submission is not received by the Grace Date.

Grading: Coding Style 5 pts, Test Cases 95 pts, Total 100 pts.

Assignment Questions (95 pts)

1. Implement a default constructor. By default, the size of the hash table should be 11 and p, one of the constants for polynomial rolling hash functions, should be 31.

2. Implement a parameterized constructor taking both the size of the hash table and p as arguments.

3. Implement accessor methods for **size**, **numElements**, and **p**.

4. Implement the **hash** method. Here, we will use a polynomial rolling hash function. In particular, for a string s of size l,

$$hash(s) = \left(\sigma(s[0])p^0 + \sigma(s[1])p^1 + \cdots + \sigma(s[\ell-1])p^{\ell-1}\right) \bmod m$$

where σ(c) is the ASCII value of the character c and m is the size of the hash table. Try using both an int and an unsigned int to keep track of the hash value. Are the results different for some longer words? Why might this be the case?

5. Implement the insert method. New elements should be added at the end of chains.

6. Implement the search method. If the target value is not in the table, return -1. Otherwise, return its location in the hash table.

7. Implement the remove method. If multiple copies of a value exist in a chain, the first copy should be removed.

8. Implement the resize method. Values should be rehashed starting at the beginning of the underlying vector and working through chains beginning to end.