CS-202
DATA STRUCTURES

PA 4
Hashing

Due Date - 11:55pm on Sunday 28th November 2021

Welcome to the fourth programming assignment of this course.
In this assignment you will be required to implement a Hashing Class through 2 techniques:

- Chaining
- Linear Probing

For submission, please zip all files in a folder and you are required to submit the assignment using the following convention:

PA4_rollnumber.zip

Note:

The course policy about plagiarism is following:

⇒ All submissions are subjected to plagiarism detection.
The teaching staff will be using MOSS.

⇒ Please be aware of the following points:

- Students cannot copy any code from the Internet.
- Students should not share their code with other students.
- If assistance received in any part of the assignment, students should indicate that specific part of code.

⇒ All students should be able to explain any part of the code they submit, if questioned.

Part 1:

Hashing Function

In the first part, you will be implementing the bitwise functions in the hashfunctions.cpp file.

The hash code function to be implemented is described as below:

bitwiseHash

Initialize **bitwiseHash**= 0
For every s_i in str
bitwiseHash = **bitwiseHash ^ ((bitwiseHash**<< 5) + (**bitwiseHash**>> 2+) s_i)

Functions:

1. unsigned long **bitwiseHash (**string value)

   The function takes a string 'value' and calculates a hash key using the above hash operation.

2. unsigned long **divCompression** (unsigned long hash, long tableSize)

   The function takes a hash key and compresses it to fit into the size of the hash table.

   tableSize is the size of array maintained by the hash table.

Part 2:

Chaining

In the second part, you will be implementing the hash table using chaining to resolve the functions. To implement chaining, you will be using the LinkedList implementation given.

Note: The hash table will have a fixed size.

All member functions listed below are declared in the Chaining.hpp file. Code to implement this part will be written in Chaining.cpp file.

Member Functions:

1. HashC (int size)

   Constructor that creates a hash table of given size.

2. unsigned long **hashkey** (string input)

   Function returns a hash key corresponding to input string. Use hash functions implemented in the Task 1.

3. void **insertWord** (string word)

   Insert given string ‘word’ into hash table.

4. void **deleteWord** (string word)

   If the given string ‘word’ exists in the hash table, deletes string.

5. ListItem<string>* **lookupWord** (string word)

   Locates 'word' in the hash table and returns a pointer to it. If string ‘word’ does not exist, return NULL.

* You may declare any helper functions, but the declared functions should NOT be altered.

For testing the implementation of this part, code and run:

Compile: g++ test_chaining.cpp -pthread -std=c++11

Run: ./a.out

Part 3:

Linear Probing

In the second part, you will be implementing the hash table using open addressing with linear probing.

All member functions listed below are declared in the LinearProbing.hpp file. Code to implement this part will be written in LinearProbing.cpp file.

Member Functions:

1. **HashL** (int size)

   Constructor that creates a hash table of given size.

2. unsigned long **hashkey** (string input)

   Function returns a hash key corresponding to input string. Use hash functions implemented in the Task 1.

3. void **resizeTable**()

   Resize the hash table when the load factor becomes too large

4. void **insertWord** (string word)

   Insert given string ‘word’ into hash table.

5. void **deleteWord** (string word)

   If the given string ‘word’ exists in the hash table, deletes string.

6. ListItem<string>* **lookupWord** (string word)

   Locates 'word' in the hash table and returns a pointer to it. If string ‘word’ does not exist, return NULL.

* You may declare any helper functions, but the declared functions should NOT be altered.

For testing the implementation of this part, code and run:

Compile: g++ test_linearprobing.cpp -pthread -std=c++11

Run: ./a.out

Marks distribution:

Part	*Marks*
Hashing function	10
Chaining	30
Linear Probing	40
Viva	20

The programming assignment holds a weightage of total 80%. After the end of the assignment, each student will be giving a viva related to the programming assignment that will hold a weightage of total 20%.

Submission Policy:

* ZIP all the files and name the folder as PA4_rollnumber.zip

*Start the assignment early!

Happy Coding:)