CS2413 – Spring2022: Bonus Project

Due on 11:59PM, 04/30/2022

***Objective:*** In this project, we shall learn about Hash Tables; insert, remove & hashing operations.

***Introduction:***

Hash tables are a great way to store and search elements, where you have a pair of elements (key, value) for each entry into the hash table. To store a value, we find the key for a given value to be inserted using a hash function.

In the project, the values to be stored into hash function are strings (in our case names). The key for each of these names will have to be calculated using the following has functions.

*h(x) = x[0] – ‘A’*

Here, *h(x)* is the hash function on a string *x*. This function will result in an integer *i* which is the

key or the index position where the name must be stored in a hash table.

For example:

1. The name to be stored is ‘Dakota’.
2. So the key = *h(x) = x[0] – ‘A’*

🡪 *h(x) = ‘D’ – ‘A’*

🡪 *h(x) = 68 – 65 = 3*

🡪 Key = 3

1. Here the key indicated the index of the hash table to where the value to be inserted.
2. Note that in C and C++, chars are identified with their numeric codes (ASCII) and hence numeric operators can be applied on them.

NOTE: You may now wonder, what if two names have the same key? For example, ‘Cameron’ and ‘Cairo’ both have the key 2. This is referred to a *collision*. There are many ways in which you can handle collision.

The method that we are going to adopt here for this project is called *chaining*. Visiting our example of ‘Cameron’ and ‘Cairo’ again, first we store ‘Cameron’ at index (key) position 2. When we get ‘Cairo’ next, we *chain* it behind ‘Cameron’ in the same index position of 2.

For the implementation of the hash table, we look into vectors data structures. The STL library for vector will be imported. Our objective in this project is to store the given data (names) in a hash table using vectors.

***Input explanation:***

A sample input file is given. There are three commands you can expect.

* ***I Cameron***
  + ‘I’ for insert followed by the name to insert.
* ***R Cameron***
  + ‘R’ for remove followed by the name to remove.
  + The name to be removed, if not found must display appropriate message given in the sample output file.
* ***D***
  + ‘D’ for displaying the entire hash table in the format given in the output file.

A sample output file is also provided.

***Class definition:***

* ***template\_bonus.cpp :*** This contains the class where you implement the hash table using a 2D vector.
* You will implement the methods that are given in this file.

***Rubrics:***

* Insert : 60 points
* Remove : 60 points
* Display : 20 Points
* Documentation : 10 points

***Instructions:***

1. All code must be written in standard C++ (so that it can be compiled by a g++ compiler).
2. You must submit: A file named **Bonus\_CS2413.cpp** that contains all the code of this project.
3. You are provided: A file named template\_bonus.cpp including the template your submission should follow, as well as sample input and output files of your program. These files will be available in Module “Projects” in Canvas.
4. All input will be read via redirected input. That is, you should not open a file inside the program.
5. Please refer to the sample input and output files given, and make sure to follow these formats exactly. The classes structure should be as shown in the provided template file.

Redirected Input:   
Redirected input provides you a way to send a file to the standard input of a program without typing it using the keyboard. Please check out the Visual Studio Installation and Setup guidelines for C++.doc provided to you on canvas. Section 3 in the document has instructions on setting up the redirected input to Visual Studio.