CSE310 Project 1, Phase 1

Due: Thursday, October 13th, **8:00pm**
On-line submission, No late submission will be accepted
Important: This is an individual project. Please do not collaborate.

Make sure that you write every line of your code. Using code written by someone else will be considered a violation of the academic integrity and will result in a report to the Dean's office.

Minimal Submitted Files

You are required to turn in the following source file(s) in C++:
Project1P1.cpp
containing a main function.
You can also submit additional files.

Objectives:

This is to practice making hash tables.

Program Description

You will develop a C++ program that reads an input data set consisting of four parts: the first part is a hash table size requested by a user; the second part is a list of athletes with a medal, this part will end with the line "InsertionEnd"; the third part is a number of commands to follow; and the forth part is a list of commands.

1. After reading in a hash table size requested by a user, a hash table of the size (with chaining) needs to be created. Each slot of your hash table should be a link list of nodes where each node represents one athlete. Initially all linked lists should be empty.

2. Then by reading each athlete information line by line, their information needs to be stored in the hash table using a hash function. You will need to design your hash function so that it reduces the number of collisions, i.e., the length of each linked list should not be too long.

Each athlete data will be in one line and will contain: discipline, gender, team_or_ind, event, venue, medal, athlete, and country that are separated by commas.
The following shows an example of such data for athletes:

Archery,M,TEAM,Archery,Lord's Cricket Ground,1. GOLD,team ITA,Italy
Archery,M,TEAM,Archery,Lord's Cricket Ground,2. SILVER,team USA,United States
Archery,M,TEAM,Archery,Lord's Cricket Ground,3. BRONZE,team KOR,South Korea
Cycling,M,IND,Road,Regent's Park,1. GOLD,Aleksander Winokurow,Kazakhstan
Cycling,M,IND,Road,Regent's Park,2. SILVER,Rigoberto Uran,Colombia
Cycling,M,IND,Road,Regent's Park,3. BRONZE,Alexander Kristoff,Norway
Fencing,F,IND,Foil,ExCeL,1. GOLD,Elisa Di Francisca,Italy
Fencing,F,IND,Foil,ExCeL,2. SILVER,Arianna Errigo,Italy
Fencing,F,IND,Foil,ExCeL,3. BRONZE,Valentina Vezzali,Italy

3. After the line “InsertionEnd”, a user will enter a number of commands.

4. Each command will be “hash_display”, “hash_search”, or “hash_delete”.

Hash_Display command:
With “hash_display” command, your program needs to display the content of your hash table by listing the content of each linked list in the following format, by specifying the index of each slot of the hash table and their linked list size (if a linked list is empty, it should print out “*The list is empty”*:

index: 0, linked list size: 3

discipline: Cycling
gender: F
team_or_ind: IND
event: Road
venue: Regent's Park
medal: 1. GOLD
athlete: Marianne Vos
country: Netherlands

discipline: Swimming
gender: F
team_or_ind: IND
event: Medley
venue: Aquatics Centre
medal: 3. BRONZE
athlete: Xuanxu Li
country: China

discipline: Fencing
gender: F
team_or_ind: IND
event: Foil
venue: ExCeL
medal: 1. GOLD
athlete: Elisa Di Francisca
country: Italy

index: 1, linked list size: 2

discipline: Swimming
gender: M
team_or_ind: IND
event: Freestyle
venue: Aquatics Centre
medal: 2. SILVER
athlete: Tehwan Park
country: South Korea

discipline: *Judo*
gender: *M*
team_or_ind: *IND*
event: *60*
venue: *ExCeL*
medal: *3. BRONZE*
athlete: *Rishod Sobirov*
country: *Uzbekistan*

index: 2, linked list size: 2

discipline: *Cycling*
gender: *M*
team_or_ind: *IND*
event: *Road*
venue: *Regent's Park*
medal: *3. BRONZE*
athlete: *Alexander Kristoff*
country: *Norway*

discipline: *Swimming*
gender: *M*
team_or_ind: *IND*
event: *Freestyle*
venue: *Aquatics Centre*
medal: *3. BRONZE*
athlete: *Peter Vanderkaay*
country: *United States*

index: 3, linked list size: 0
The list is empty

index: 4, linked list size: 1

discipline: *Swimming*
gender: *F*
team_or_ind: *IND*
event: *Medley*
venue: *Aquatics Centre*
medal: *2. SILVER*
athlete: *Elizabeth Beisel*
country: *United States*

index: 5, linked list size: 1

discipline: *Judo*
gender: *F*
team_or_ind: *IND*
event: *48*
venue: *ExCeL*
medal: *1. GOLD*
athlete: *Sarah Menezes*
country: *Brazil*

Hash_Search command:
Hash_Search command will have the format of:

hash_search,discipline,gender,event,athlete

where the word "hash_search" is followed by discipline, gender, event and athlete, and they are separated by commas.
A real example of such command can be:

hash_search,Shooting,F,Pistol,Jongoh Jin

After the hash_search command is entered, the program should search for an athlete that matches those data fields, and if it is found, display their information with their medal type as:

The medal recipient Jongoh Jin has the medal of 1. GOLD

If not found, display a message using the following format:

Jongoh Jin for Shooting with event Pistol not found

Hash_Delete command:
Hash_Delete command will have the format of:

hash_delete,discipline,gender,event,athlete

where the word "hash_delete" is followed by discipline, gender, event and athlete, and they are separated by commas.
A real example of such command can be:

hash_delete,Swimming,M,Medley,Thiago Pereira

After the hash_delete command is entered, the program should search for an athlete that matches those data fields, and if it is found, it should be deleted from the hash table and the program needs to display a message using the following format using their information:

The medal recipient Thiago Pereira for Swimming with event Medley deleted

If not found, display a message using the following format:

Thiago Pereira for Swimming with event Medley not found

Design Requirements:

You should create a hash table with chaining (an array of linked lists). Please specify your hash function *h(k)* clearly. The key for each athlete data will be a string made by appending their discipline, gender, event, and athlete. For instance, for the athlete with the information:
"Cycling,M,IND,Road,Regent's Park,1. GOLD,Aleksander Winokurow,Kazakhstan",

the key will be *"CyclingMRoadAleksander Winokurow "*.
You also need to define INSERT,SEARCH, and DELETE functions for the hash table, and define INSERT,SEARCH, and DELETE functions for the linked list. Please have comments to clearly identify these functions in your code.

Implementation/Documentation Requirements:

- You need implement this program using C++ and it has to read from the standard input (from a keyboard).
- Your program needs to compile and execute in general.asu.edu.
- You need to define Insert, Search, and Delete functions (name them "hash_insert", "hash_search", and "hash_delete") for your hash table, and a hash function *h*.
- Your code should be well documented and commented.
- You also need to defined Insert, Search, and Delete functions in your linked list.
- At the top of your driver file, within a comment, you need to write your hash function by specifying h.
- **At the top of your driver file, within a comment, write your hash analysis -- how did you come up with your hash function? Was the number of collisions what you expected? Did your hash function work well? If you had to change your function to reduce the number of collisions, how did you change it?**
- You must use the provided data sets.
- Also you are not allowed to use any predefined data structures (such as ones in the library in C++, etc.) except arrays and strings, you need to build your own data structures and operations associated with them (such as insert or search).
- Copying any codes from other people's programs is considered to be cheating and will lead to a report to the Dean and you will be penalized. Also check the rules stated in the syllabus of the course regarding the academic integrity

What to turn in:

Under the submission link on Blackboard, submit a zip file named yourFirstName-yourLastName.zip containing the following:

1. one or more files of your well documented C++ source code (with a file extension .cpp or .h) implementing the commands required, your hash table, and linked list.

2. A Makefile that compiles your program to an executable named "p1p1" on the Linux machine general.asu.edu. Our TA will write a script to compile and run all student submissions on general.asu.edu; therefore executing the make command must produce the executable file "p1p1" in the same folder as your source code files. After such p1p1 file is produced, then we should be able to execute your program using the command:

./p1p1 < sampleinput.txt

where "sampleinput.txt" is a file name, but we should be able to use a file with a different name with a content in the format specified in the project statement.

Your program must read from a keyboard, not a file.

Error Handling

Your program will be tested with other input besides the sample input given, thus is expected to be robust.

Grading Criteria

2 pts – Documentation (correct and complete code description and comments, header for each function/method)
1 pt – Indentation and Spacing (easy to read)
1 pt – "makefile" is submitted and it works correctly to generate p1p1 executable file
4 pts – Linked List class or struct is defined with insert, search, and delete function
4 pts – Hash Table class or struct is defined with hash_insert, hash_search, and hash_delete function
2 pts – Hash function is defined with Hash Table and data are hashed uniformly
2 pts – Hash Analysis (It should be written at the top of your driver program as a comment)
14 pts – Correct output for the test cases

Total points: 30