CS 311 - HW 10 - 100 points

Hash function

Steps:

1. For this HW, you need the implementation of llist and slist from HW5 and HW6. Make sure that your slist works correctly. Copy .h and .cpp files into your HW11 folder.
2. Copy the attached elem.h, elem.cpp, htable.h, htable.cpp and hw11client.cpp to your folder.
3. Each element of a linked list node has to have a key (a 3 digit account number) and a name (a string). We need to make this change quickly without changing llist and slist that much. This HW shows how easy it is to change el_t so that it contains multiple parts. We need to define el_t in elem.h and elem.cpp to do this. PLEASE STUDY the files very carefully!
4. Study elem.h and elem.cpp, which will be used by the client to define el_t.
5. Update your linked list classes
   - #include elem.h
   - Copy search as search2 and update search2 so that it returns the found element itself (el_t) instead of the position. If not found, it returns a completely blank el_t.
6. Use the attached htable.h (remains unchanged) to create htable.cpp.
   Data Member:
   an array containing slists**.** Table size is 37.
   Functions:
   int hash(int key) turns the key into a slot number using % table size. Private.
   int add(el_t) adds the element to the table and return the slot number.
   add will have to call your private hash function to convert the key into a slot number.
   el_t find(int key) finds the element given the key. Need to call the hash funciton.
   void displayTable() displays the table in a readable format, including slots numbers.
7. Refer to the sample results given at the end of this file to see how the output format should look like. Note that the sample adds only a few keys. You must add almost 20 keys.

8. Complete the client file: **hw11client.cpp**
   The file should contain three steps as follows.
   1. Add: Use a loop to add almost **20** names to the table. Make sure some of them collide.
   2. Call displayTable.
   3. Lookup: use a loop to lookup keys in the table. If found, display both key and name.
      - Check four keys that **are** in the table.
        - One with no collision, the others with collisions. One is the first node in the linked list and two are the next nodes.
      - Check two keys that are **not** in the table.

9. Run the program and copy the results into **Test1.txt**
   The format must match the sample results given at the end of this file.

Submission

Submit a zip file containing the following files.

1. .h and .cpp files for llist and slist -- copy them from HW5 and HW6 and
   - Add search2() function
   - Change the llist el_t to an elem (remove typdef)
2. elem.h and elem.cpp -- copy them from the attached files
3. htable.h -- class header file (remains unchanged)
4. htable.cpp (50 points) -- class implementation file (complete the file)
5. HW11client.cpp (50 points) -- the implemented application (Hashing program)
6. Test1.txt -- copy the results into this file

Note that you must enter almost 20 keys into the hashing table.

Important note1: You will miss up to 10 points if you don't comment your programs.

Important Note2: Always make sure the files you submit can be compiled on **empress.csusm.edu** with no error. We will compile and test your files on empress.

Sample result

```
0:[empty]
1:[empty]
2:[empty]
3:[empty]
4:[empty]
5:[empty]
6:[empty]
7:[empty]
8:[empty]
9:[empty]
10:[empty]
11:[empty]
12:[empty]
13:[empty]
14:[empty]
15:[200+John 237+Jeremy ]
16:[201+Steve 238+Nancy ]
17:[empty]
18:[empty]
```

```
19:[empty]
20:[empty]
21:[empty]
22:[empty]
23:[empty]
24:[empty]
25:[empty]
26:[100+Carlo ]
27:[101+Mary ]
28:[empty]
29:[empty]
30:[empty]
31:[empty]
32:[empty]
33:[empty]
34:[empty]
35:[empty]
36:[empty]
Look for? 200
Found 200: 200+John
Look for? 238
Found 238: 238+Nancy
Look for? 99
99 not found.
Look for? 30
30 not found.
```