DSA 5005 Data Structures – Summer 2017 – Programming Project 2
Due July 24, 2017 – 11:59 PM

Objectives: To learn and manipulate hash tables, vectors, and linked lists. Use STL libraries.

Description: The input consists of a set of strings on each line. The goal is to perform a *hash* on each string and store them in a *vector of linked lists* data structure. You will construct a HashTable class that has the vector of linked lists data structure in it. The hash function *h* on a string *x* denoted *h(x)* results in an integer *i* which is an index position of the vector of linked lists. For example, if the hash function is *h(x) = x[0] – 'A'*, then for the string *"Davis Joe"*, the function value would be *3*. Note that in C and C++, chars are identified with their numeric codes and hence numeric operators can be applied on them. The numeric value of character *D* is 'D' – 'A'.

For your project, you will initially start with a *vector linked lists V* initialized to being empty. Note that in V, each element points to a linked list. Next, you process each string *x*, by adding the string *x* in the linked list pointed by *V[h(x)]*. For example, if *x* is "David Joe", then the string "David Joe" is added to the linked list pointed by *V[3]*. **If the hash function gives an index position that is not currently in the vector, then the vector should be expanded to accommodate that position. Initially the size of the vector should be 0.**

Your program should be able to handle the following operations.

1. find(*x*), determine if the string *x* is in *V*. For this purpose, apply the hash function on *x*, get the index position *i* and search the linked list associated with *V[i]*. If the element is not found, search in the linked lists *V[i+1], V[i+2], V[i+3]*, and so on. Note that you should not search the linked lists above *V[i]*.
2. split(*i*, *p*), if the linked lists *V[i]* has more than *p* elements, then keep the first *p* elements in *V[i]* and move the rest **one at a time** to positions below *i* if they have less than *p* elements, otherwise move them further down. The vector might have to be expanded during this operation.
3. remove(*x*), find and delete the string *x* from *V*.
4. insert(*x*), apply the hash function and insert the string *x* in *V*. The vector might have to be expanded during this operation.
5. The HashTable class that you will be developing should have the following methods apart from the methods 1-4 above: empty constructor, destructor, and ostream operator

You will receive a 20% bonus if you write the copy constructor and an overloaded = operator.

Use can use the STL library for vectors and linked lists. Examples of such usage are given in the appendix of your textbook. You should use a character array to represent a single string. **You can test your program with a main program and I will provide another main program later**.

Good Luck!

```
void main()   /* A sample main program */
{
  HashingTable<char*>* myHashTable;
  char* Names[25] = {"Andy B", "Amy Dean", "Antonio G", "Andy Roberts",
                     "Brian W","Bob Macy", "Brent James", "Buck Muck",
                     "Cannon James", "Cart Wright", "Catherine Zeta", "Carl Lewis",
                     "David Johnson", "Dowd Peter", "Daniel Fauchier", "Dawn Smith"
                     "Yarda Varda", "Yami Jacob", "Yester Louis", "Yukon Oklahoma",
                     "Zandi Rich", "Zea John", "Zelby Leon", "Ziert Paul", "Zanola Helen"};
  int i;

  myHashTable = new HashingTable<char*>(26);
  for (i=0; i < 26; i++)
        (*myHashTable).insert(Names[i]);

  cout << "Printing the hash table after inserting...." << endl;
  cout << myHashTable << endl;

  if ((*myHashTable).find("Zandi Rich"))
        cout << "Zandi Rich is in the list" << endl;
  else
        cout << "Zandi Rich is not in the list" << endl;

  (*myHashTable).remove("Zea John");
  if ((*myHashTable).find("Zea John"))
        cout << ""Zea John is in the list" << endl;
  else
        cout << "Zea John is not in the list" << endl;

  for (i=0; i < 26; i++)
        (*myHashTable).split(i,2);

  cout << "Printing the hash table after splitting...." << endl;
  cout << myHashTable << endl;

  if ((*myHashTable).find("Ziert Paul"))
        cout << "Ziert Paul" is in the list" << endl;
  else
        cout << "Ziert Paul" is not in the list" << endl;

  (*myHashTable).insert("Zea John");
  if ((*myHashTable).find("Zea John"))
        cout << ""Zea John is in the list" << endl;
  else
        cout << "Zea John is not in the list" << endl;

  delete myHashTable;
}
```