CSCI 340 **Computer Assignment 8** **Spring 2018**
(10 points)

Hashing with Linear Probing

For this computer assignment, you are to write a C++ program to create, search, print, and manage an item inventory. The program is partially implemented in assignment8.h and assignment8.cc, which are available at `/home/turing/mhou/public/csci340spring2018`.

The item inventory information will be kept in a hash table. Each table entry is described by the following structure, as specified in the header file `assignment8.h`.

```
struct Entry {
    string key;
    string description;
    Entry() { key = "---"; }
};
```

The `key` field is the item identifier, which has two uppercase letters followed by a decimal digit (e.g., `AD5` or `XR8`). The `description` field contains the item description. There is a default constructor to create an empty entry, where the key is "---".

The item inventory system is defined as a class, named as `HT`, and its definition is given in `assignment8.h`. The hash table `hTable` (container of type `vector < Entry >`) is used to store the entries of the item inventory. `table_size` is the size of the hash table. `item_count` records the number of items currently in the system.

The item table can be accessed directly using the *linear probing* technique to resolve the collisions. The hash function expects the `key` of an item as the input argument, and it returns an integer in the range `[ 0 … (table_size-1) ]`. The function has the following prototype: `int HT :: hash ( const string& key )`. The source file assignment8.cc contains the implementation of this function.

The member functions of `HT` are described below:

- `HT :: HT ( int s=11 )` : This is the constructor. The hash table `hTable`, which is a vector of `Entry`, is created dynamically (with `new` operator) for a given size `s` (with default value 11) here. Other data members are also properly initialized here.

- `HT :: ~HT ( )` : This is the destructor. Since the hash table `hTable` is implemented as a dynamically created vector, its memory is released here by `delete` operator.

- `int HT :: search ( const string& key )` : This *public* member function searches the hash table for a record with a given `key`. It first of all needs to compute the hash value of the given key. If the search is successful, this function returns the position of the found item; otherwise, it returns -1.

- `bool HT :: insert ( const Entry& e )` : This *public* member function inserts item `e` in the hash table. It first of all checks if the item's `key` already exists in the table or not. If yes, this function prints an error message and returns false. If the hash table is already full, this function also prints an error message and returns false. It then needs to compute the hash value of item `e`'s key. If the hash table position of the hash value is empty, the item is placed on that positon; otherwise, an appropriate position is determined by linear probing. After the item is inserted, `item_count` is increased and the function returns true.

- `bool HT::remove(const string& s)` : This public member function removes an item with given key `s`. It first of all searches the given key in the hash table. If the key is not found, this function simply returns `false`; otherwise it removes the entry of the given key by setting its key as "+++", decreases `item_count`, and returns `true` at the end.

- `void HT :: print ( )` : This *public* member function prints the existing entries in the hash table: the index value of the position, the key, and the description.

The input file `assignment8input.txt` contains operations on the item inventory. Each line specifies an operation. There are four types of operations, as described below:

- Insertion operation is specified by the format
  `A:item-key:item-description`
- Deletion operation is specified by the format
  `D:item-key`
- Search operation is specified by the format
  `S:item-key`
- Print operation is specified by the format
  `P:`

The driver program reads the input file, and conducts proper operations based on the first character of each line. The main function is completely implemented in assignment8.cc. You are required to implement two global functions to parse above operations:

`Entry* get_entry (const string& line)` : This method takes a line of input and parses it to create a new Entry. The line is in format of `A:item-key:item-description`, i.e., three strings separated by `':'`, where the first string is just a single character `'A'`.

`string get_key (const string& line)` : This method takes a line of input and parses it to return the item-key. The line is in format of `X:item-key`, i.e. two strings separated by `':'`, where the first string is just a single character.

Programming Note:

- The records stored in the inventory system have unique keys. Duplicate keys are not permitted.
- Initially an empty position (a vector element `Entry`) contains key "---" (See the constructor of `Entry`.) When an entry is deleted at a position, that position then contains key "+++" (See the `remove` function of `HT`.) Both cases indicate a position is empty and can save a new entry. But for the search operation, you need to treat two cases differently due to the nature of linear probing technique.
- Include any necessary headers and add necessary global constants.
- You are not allowed to use any I/O functions from the C library, such as `scanf` or `printf`. Instead, use the I/O functions from the C++ library, such as `cin` or `cout`.
- In the final version of your assignment, you are not supposed to change existing code, including class definition and the main method, provided to you in the original files `assignment8.h` and `assginment8.cc`. You can insert new code to both files. Usually the source file (.cc) contains the implementation of member methods of the class(es), together with global functions. You can add inline code in the header file (.h) when the implementation is very brief, containing just one or two lines for a member method.
- To compile the source file, execute " `g++ -Wall assignment8.cc –o assignment8.exe`". This will create the executable file `assignment8.exe`. To test your program, execute "`./assignment8.exe assignment8input.txt &> assignment8.out`", which will put the output (including any error messages) in file `assignment8.out`. You can find the correct output of this program in file `assignment8.out` in the directory shown in the last page. `assignment8input.txt` is also in that directory.
- Add documentation to your source file.
- Prepare your `Makefile` so that the TA only needs to invoke the command "`make`" to compile your source file and produce the executable file `assignment8.exe`. Make sure you use exactly the same file names specified here, i.e. `assignment8.cc` and `assignment8.exe`, in your `Makefile`. Otherwise your submission will get 0 points.
- When your program is ready, submit your files `assignment8.h`, `assignment8.cc` and `Makefile` to your TA by following the Assignment Submission Instructions.