ECE 36800 – Data Structures
Programming Assignment 3 – Part 1

Purpose of Assignment 3:
Design and implement a hash table to store words and count their occurrences in a text file.

Goal of Part 1:
Implement a chained hash table and a quadratic probing based hash table to store key words.

What to submit:

1. `table1.cpp`: The implementation file for the `ChainTable` and `QuadTable` classes. Start with the provided file and add your implementation.
2. A word document `proj3p1.docx`: This file should include the printout of your program with `keys.txt` as the input text file. Make sure you test both menu options. Include a copy (or screenshot) of the printout of the program run.
3. Push all your files under the "proj3/part1" directory before the deadline.

Other available files (please do not modify):

1. `table1.h`: Header file for `Table` base class and derived `ChainTable` and `QuadTable` classes
2. `keys.txt`: A text file including a list of key words: Each line afterwards contains one single key word. Note that there may be duplicate words in the file.
3. `tabletest1.cpp`: A simple test program to insert key words into the selected hash table.
4. `Makefile`: Rules to compile the source files for your convenience.

Guideline:

Based on user choice, the `tabletest1.cpp` program creates either a `ChainTable` object or a `QuadTable` object called `keytable`. The program then reads one key word at a time from `keys.txt`, calls the corresponding `insert` function to add the key word into `keytable`, if the key word is not already in the table.

Explanations of Implementation:

- `Table` is an abstract base class. It records of the actual number of strings stored in the table. It also provides a `hashcode` function to calculate an index in the table based on a `string` type input value. For a list of data types that comes with a default hash function, please refer to http://www.cplusplus.com/reference/functional/hash/.
- `Table` also provide two pure virtual functions `insert` and `print`. These two functions are meant to be overridden in the derived classes.
- `ChainTable` is derived from base `Table` class. In a `ChainTable` object, the actual records are stored in a chained hash table. `datatable[TABLE_SIZE]` is an array, with each entry storing a list of strings. The `list` container class in the C++ STL is

used here. You can explore the `list` container class and its member functions in C++ STL at http://www.cplusplus.com/reference/list/list/.

- `QuadTable` is also derived from base `Table` class. In a `QuadTable` object, the actual records are stored in an array. `datatable[TABLE_SIZE]` is an array, with each entry storing one string. `QuadTable` has a member function `full()` to test if the `datatable` is full or not. In your implementation of the `insert` function, please use the simple probe function $c(i) = i^2$, i=1, 2, ..., to find the relocation distance when a collision occurs.
- `keys.txt` should be used as a command line argument.

Grading Policy:

Part 1 counts for 40% of the overall points in Programming Assignment 3.

Please make sure your program compiles successfully. If not, 20 points will be deducted.

1. **Executability** (5%):
   - Runtime errors: You program must not have runtime errors (e.g., code crash, infinite loop, reading uninitialized memory, accessing the content of a NULL pointer, etc.).
2. **Programming style** (5%):
   - Code efficiency: Code should use the best approach in every case.
   - Readability: Code should be clean, understandable, and well-organized. Please pay special attention to indentation, use of whitespace, variable naming, and organization.
   - Documentation: Code should be well-commented with file header and comments.
3. **Program Specifications/Correctness** (30%):

   Please refer to the Grading Criteria table for details. Specifically, your program should behave correctly, adhere to the instructions, and pass the test program.

file	item	weight (%)
table1.cpp	`ChainTable::insert`	10
	`QuadTable::insert`	10
proj3p1.docx	Running result	10
	total	**30**