ECE 36800 – Data Structures
Programming Assignment 3 – Part 2

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

Goal of Part 2:
Implement a chained hash table and a quadratic probing based hash table to store words and their count. Find the word with the highest count.

What to submit:

1. `table2.cpp`: The implementation file for the `ChainTable` and `QuadTable` classes. Start with the provided file and add your implementation.
2. A word document `proj3p2.docx`: This file should include the printout of your program with `alice.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/part2" directory before the deadline.

Other available files (please do not modify):

1. `table2.h`: Header file for `Table` base class and derived `ChainTable` and `QuadTable` classes
2. `tabletest2.cpp`: A simple test program.
3. `alice.txt`: A text file including the whole "Alice's Adventures in Wonderland".
4. `Makefile:` rules to compile the source files for your convenience.

Guideline:

Based on user choice, the `tabletest2.cpp` program creates either a `ChainTable` object or a `QuadTable` object called `keytable`. The program then reads from an input text file, and calls the corresponding `insert` function to add the words into `keytable`. If a word is already in the table, increase the `count` by 1. So that the `count` associate with each record reflect the number of occurrences of that word in the file. In the end, the program prints out the word that appears most frequently in the input text file. The `count` of that word is also printed.

Explanations of Implementation:

- `Table` is an abstract base class. It records of the actual number of records stored in the table. It also provides a `hashcode` function to calculate an index in the table based on a `string` type input value.
- `Table` also provide two pure virtual functions `insert` and `print_max`. These two functions are meant to be overridden in the derived classes.

- `Record` is a user defined class storing two member variables: `string key` and `unsigned int count`.
- `ChainTable` is derived from base `Table` class. the actual records in a `ChainTable` object are stored in a chained hash table. Unlike in part 1, `datatable[TABLE_SIZE]` is an array, with each entry storing a list of `Record` objects.
- `QuadTable` is also derived from base `Table` class. In a `QuadTable` object, the actual records are stored in an array. Unlike in part 1, `datatable[TABLE_SIZE]` is an array, with each entry storing one `Record` object.
- `alice.txt` should be used as a command line argument.
- `tabletest2.cpp` reads one line at a time from the input file, then extract each word separately. Before insert the word into the hash table, it first calls the transform function to convert the word into all lower case. So the counting of words is case insensitive, for example, 'The' and 'the' are considered the same word. Also note that punctuation will be included in extracting strings. For example, "me" and "me." and "me?" and so on, are considered different words and should be inserted as different entries in the hash table.

Grading Policy:

Part 2 counts for 60% of the overall points in Programming Assignment 3.

Please make sure your program compiles successfully. If not, 30 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** (50%):

   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 (%)**
table2.cpp	`ChainTable::insert`	10
	`ChainTable::print_max`	10
	`QuadTable::insert`	10
	`QuadTable::print_max`	10
proj3p2.docx	Running result	10
	total	**50**