Assignment #5
Hashing
PROG 2400: Data Structures
As shown for your Section in D2L

Task:
This assignment looks at hashing and collision resolution.

Part 1: Hashing
Create a console application to compare the time required to conduct searching activities on using a hash table.

You are tasked to build a console-based spell checker that will be used to show the spelling mistakes in a standard text document. You should use the same program framework from assignment #4.

Requirements:

- The application will start by reading all the words in a dictionary file that has been provided for you. The file is not a complete dictionary. It simply contains all the correct spelling for specific words in alphabetic order. The dictionary words will be the basis for a custom hash table in your application.
- By programmatic means you must store each of the dictionary words in your hash table. When the table is complete you must display it for review. You must consider the data being stored and devise your own hashing function and collision resolution mechanism. The suggest approach is to use some form of modular arithmetic combined with collision resolution by chaining, but ultimately it is your choice. Be prepared to fully explain your choice and algorithm.
- Reusing the code from part 2 of the binary search tree assignment, create the infrastructure to conduct the searching portion of the spell check, replacing the binary search tree with your hashing table and function.
- Once the hash table has been filled, you will read in a second file that is a sample document needing spell checking. You will then compare each word in the document against the hashed words.
    - If the search fails, you will display the misspelled word to the console.
- **Hint**: You cannot modify either the provided dictionary file or document file. To simplify the timing capture, consider reading each word of the target file to spell check in to a simple array and use the array to source the searches. Then use a second array to capture the misspelled words for later display. This should provide the needed isolation.

Assignment #5
Hashing
PROG 2400: Data Structures
As shown for your Section in D2L

Evaluation:

This assignment is worth 31 marks. Please see the marking rubric below.

Assignment Notes:

The assignment must be demonstrated to the instructor on or before the due date during class.

If your assignment is late please send an e-mail to the instructor, hal.o'connell@nscc.ca, to confirm submission. This e-mail will constitute the timestamp for evaluating any late penalty the assignment may incur.

See the **Marking Rubric** below.

Assignment #5
Hashing
PROG 2400: Data Structures
As shown for your Section in D2L

Criteria	Marginal	Developing	Good	Exceptional	Marks
	0	1	2	3	
Hash Algorithm	• Hash table not used to store dictionary • too many errors exist	• Dictionary was correctly read into hash table - can be viewed on the console • Some errors exist	• Dictionary hash table was read and displayed correctly • Few errors exist	• Hash table contains all dictionary words and all spelling errors identified • Hash algorithm can be explained - can easily view table and any chains on the console • No bugs	____**x3**_
Collision Resolution	•	• An attempt was made to resolve collisions • Some errors exist	• Collision resolution reuses earlier classes built for this course. • Few errors exist	• Collision resolution reuses earlier classes built for this course and performs without bugs	____**x2**_
				Sub Total	_____ **15**
Spell Checker Output	• The program does not output misspelled words from the test document	• Some misspelled words are correctly identified and listed • Could not handle punctuation, capitalization or special characters • Some errors exist	• All misspelled words are correctly identified and listed • No errors	•	_____
Aesthetics	• incorrect or non-existent use of whitespace in output • output is confusing and hard to follow	• fair use of whitespace • most output is clear, but poorly presented	• excellent use of whitespace • output is clear and attractively presented		_____
Readability	• source code is poorly organized and very difficult to read	• source code is fairly easy to read, but is hard to follow in some areas	• source code is exceptionally well organized and easy to follow		_____

Assignment #5
Hashing
PROG 2400: Data Structures
As shown for your Section in D2L

Criteria	Marginal	Developing	Good	Exceptional	Marks
	0	1	2	3	
Reusability	• source code cannot be reused • no functions or classes used	• portions of code could be reused with modifications	• source code could be easily reused with few modifications		____
Efficiency	• contains large portions that could have been easily reduced using a different method • too much code is replicated, copy /pasted	• tried some methods to improve efficiency • can explain what they attempted	• very clean and efficient code • can propose new ideas for improvement		____
Comments	- little to no comments used	• comments are used, some are meaningful and easily understood • some files and functions have headers	• not over/under commented • comments are meaningful and easily understood • files and functions have headers • Code is self-documenting		____
Naming Convention	• no standard naming convention followed	• a standard naming convention was used for part of the program, but deviated often	• industry standard naming convention used throughout the program		____
Consistency	• no consistency in formatting or layout of source code	• source code formatting was present but inconsistent with whitespace, brackets, etc	• source code formatting never deviated from the programmer's layout		____
				SubTotal	**16**
				Assignment Total	**31**

0 - Assignment not submitted or work not original.