# SENG1050 – DATA STRUCTURES

## FOCUSED ASSIGNMENT 3 - HASH TABLES

## OVERVIEW

- Create a program that stores words in a Hash Table and then searches for the words.

## OBJECTIVES

- Create and use a hash table.

## ACADEMIC INTEGRITY AND LATE PENALTIES

- Link to Academic Integrity Information
- Link to Late Policy

## EVALUATION

- The evaluation of this assignment will be done as detailed in the Marking lecture in Week 2 of the C course.

## PREPARATION

- Make sure that you understand how sorted linked lists work.

## REQUIREMENTS

### Input Requirements

- Create a loop in main() that gets one word at a time from the user until they enter "." as the only thing on the input line.
    - Words will not be longer than 20 characters.

### Hash Table Requirements

- In a hash table, you will keep track of words that are 20 or fewer characters long.
    - The hash table will be an array of 11 pointers to a struct called WordNode.
        - The WordNode struct that you will create will contain an array of char of length 21 (which will contain null-terminated strings) and a pointer to the next node. You can make WordNode a class if you want to.

        - You must use C-style null-terminated strings (not C++ strings).
    - Make sure that your linked lists are sorted singly-linked lists, sorted in ascending order (e.g. "alpha" would occur before "theta").
- In main, declare an array variable to contain your hash table. As mentioned previously, let the buckets (which are the array elements) contain pointers to WordNodes that would be either head pointers to a linked list or NULL.
    - The buckets are **not** WordNodes. They are **pointers** to WordNodes.

### Hash Function Requirements

- Each time you get a word, pass the word as a parameter to a hash function (called myHashFunction). This hash function must call the original djb2 function (from slide 95 of the hash tables lecture; change the parameter to work with chars rather than unsigned chars; do not use the modification mentioned on a subsequent slide) and then use the Division Method to return a value between 0 and 10 (inclusive) as a hash value.
    - Use this returned hash value in main() to determine which bucket should contain the word. Then add the word to the bucket as described in lecture.
    - To be absolutely explicit, you must have two separate functions for this. Modularity is always your friend.
        - Combining the two functions into one will result in a cap of 60.
        - Call myHashFunction() from main(). Call djb2() from myHashFunction().

### Search Requirements

- Once the user is done entering words, set up a separate loop to get input for words to search for. Then search for the words in the hash table by determining the appropriate bucket and searching the linked list found there (if there is one).
    - Every time you do a comparison in your search, use printf() (or cout) to display the word you are comparing your search word against.
    - When you finally find the word, use printf() (or cout) to display "Success!". If you ultimately don't find the word, use printf() (or cout) to display "Not there!".
    - I will be testing using lowercase only.
- This will continue in a loop until the user enters "." as the only thing on the input line.

## GIT REQUIREMENTS

- Use GitHub Classroom for revision control, similar to Focused Assignment 2. It is expected that you have a reasonable number of commits with meaningfully descriptive commit comments.

## CHECKLIST REQUIREMENTS

- Create a requirements checklist. This should contain the specific requirements from this assignment as well as any relevant requirements that have been covered in lecture or that are found in the SET Coding Standards or SET Submission Standards. Do it in

whatever form you wish. Hand in your completed checklist in PDF form as checklist.pdf. Not having this checklist will result in a cap of 80 on your mark.

## FILE NAMING REQUIREMENTS

- You must call your source file f3.cpp.
- You must call your checklist checklist.pdf.

## SUBMISSION REQUIREMENTS

- Do not hand in any other files.
- Submit your files to the *DS: Focused Assignment 3* Assignment Submission Folder.
- Once you have submitted your file, make sure that you've received the eConestoga e-mail confirming your submission. Do not submit that e-mail (simply keep it for your own records until you get your mark).