COMP 2011 Lab 7: Dictionary Using Association List

An old Sterling dictionary

Introduction

Dictionary

In this lab, a dictionary simply consists of dictionary entries. Each dictionary entry has the word and its definition.

It is implemented using association list by letting each dictionary entry be the element, with the word being the key and the definition being the value. This implies that a word appears at most once in the dictionary.

A dictionary

Linked list

Linked list diagram

Recall what you have learnt about **linked list** in the lectures. We will not go over it again here. Below is a quick diagram to help you recall.

Association list

Association list diagram

To implement a dictionary in this lab, we will use an association list. An **association list** is actually a linked list with a bit of extra features on it.

Remember that a linked list can store a list of elements of the same type. In an association list, we specify the element type further: each element is a key-value pair. That is, each element contains a key (of some type) and a value (of some type).

In an association list, a key can appear at most once. There cannot be two or more elements with the same key. However, the value can be anything.

With this, we can lookup, delete, or search an element in the association list with the key of the element only. In a linked list, you would need the entire element to perform the above operations.

The insertion behavior of an association list is slightly different. In a linked list, you can simply add an element to it. For an association list, we need to check if the key of the element to be added is already in the association list. If it is, we need to update the value of the existing element instead. Only if the key is not already present in the association list. There is also a way to deal with this. For this lab, we will go with the first option: we. Replace the old element with the same key with the new element. In more details, the old element with the same key is removed from the association list and the new element is added to the association list.

Remember that a key-value pair is a dictionary entry. In this lab, we use the association list to implement a dictionary.

If you want to know more that required for this lab, you can read more on Wikipedia. However, note the implementation this article describes is different from the one required for this lab.

C string functions

You do not need to know about string.h or manipulating C strings. So we will introduce a few functions that manipulate C strings.

All basic C string manipulation functions are available in the header string.h or cstring.h. If you are using C, it is all in the functions available and the documentation is at https://cplusplus.com/reference/cstring/. If you are using C++, it is all in the functions available and the documentation is at https://cplusplus.com/reference/cstring/. Both are equivalent.

As you can see, there are many C string manipulation functions, so we will only introduce 5 of them that you may need to use in your lab. You are not required to use any of these functions, you may also write your own. Just use what you are a practice or reading documentation. Also note that we did not test anyone using of functions, you need to learn to identify which functions are useful in your context.

The 5 functions are: strlen(), strcpy(), strcat(), strncpy(), and strncpy().

strlen()

- reference: cplusplus.com, openkroom.com

```
strlen(char* pStr);
```

Counts the C string pointed by pStr, including the null terminator, in the character array pointed by pStr.

Takes the C string as input. Returns an int that is the number of characters in the C string, including the null terminator. Or otherwise, you may not into segfault!

Examples:

```
char myStr[10];
strcpy(myStr, "LINUX");// Okay: target is 10KB
int myStrLen = strlen(myStr);// Okay: myStrLen is 5
// strcpy(myStr, "LINUXFUN");// Not okay!
```

strncpy

- reference: cplusplus.com, openkroom.com

```
char* strncpy(char* dest, const char* src);
```

Copies at most count characters from the C string pointed to by src, including the null terminator, to the character array pointed by dest.

The count includes the null terminator. If count characters have been written before encountering a null terminator in src, the rest is filled with null characters. If count characters have been written before encountering a null terminator, additional null characters are written to dest until a total of count characters have been written.

Takes the C string to be copied, and the character array to be copied to, as input. Returns a pointer to dest. Or otherwise, you may not into segfault! Make sure that the count is 1 more than the number of characters to be copied!

Examples:

```
char myStr[10];
strcpy(myStr, "LINUX");// Okay: 5
int myStrLen = strlen(myStr);// Okay: 5
strncpy(myStr, "LINUX", 5);// Okay: 5
// strcpy(myStr, "LINUX");// Okay: 5
// strncpy(myStr, "LINUXFUN", 10);// Okay: target is 10KB
strncpy(myStr, "LINUXFUN", 6);// Okay: target is 10KB, but is not a C string because it is missing a null terminator
strncpy(myStr, "LINUXFUN", 8);// Okay: target is 10KB, but is not a C string because it is missing a null terminator
strncpy(myStr, "LINUXFUN", 9);// Okay: target is 10KB, but is not a C string because it is missing a null terminator
// strncpy(myStr, "LINUXFUN", 10);// Not okay!
// strncpy(myStr, "LINUXFUN", 11);// Not okay!
```

strcat

- reference: cplusplus.com, openkroom.com

```
char* strcat(char* pStr, const char* mystuff);
```

Appends a copy of the character array pointed to by mystuff, including the null character, to the character array pointed by pStr. Or otherwise, you may into segfault!

Takes the C string to be appended, and the character array to be appended to, as input. Returns a pointer to pStr. Or otherwise, you may into segfault!

Examples:

```
char myStr[10];
strcpy(myStr, "LINUX");// Okay: 5
int myStrLen = strlen(myStr);// Okay: 5
strcat(myStr, " is");// Okay: 8
int myStrLen = strlen(myStr);// Okay: 8
// strcat(myStr, " is fun");// Okay: 13 > 10
```

strncat

- reference: cplusplus.com, openkroom.com

```
char* strncat(char* pStr, const char* mystuff, const int count);
```

Appends a copy of the character array pointed to by mystuff, including the null character, to the character array pointed by pStr. Or otherwise, you may into segfault!

Takes the C string to be appended, and the character array to be appended to, as input. Takes the number of characters from mystuff to be appended as input. Returns a pointer to pStr. Or otherwise, you may into segfault!

Examples:

```
char myStr[10];
strncpy(myStr, "LINUX", 5);// Okay: 5
int myStrLen = strlen(myStr);// Okay: 5
strncat(myStr, " is", 3);// Okay: 8
int myStrLen = strlen(myStr);// Okay: 8
// strncat(myStr, " is fun", 6);// Okay: 14 > 10
```

strncmp

- reference: cplusplus.com, openkroom.com

```
int strncmp(const char* str1, const char* str2, const int count);
```

Compares two C strings str1 and str2 lexicographically (you will see this word appear in lab 7).

Lexicographical order means comparing the characters by their ASCII value, from left to right. Returns 0 if the first count characters of str1 and str2 are identical, otherwise, it returns the difference between the first pair of characters that are not identical.

Takes two C strings str1 and str2, and the number of characters to be compared, as input.

Returns an int.

Examples:

```
char myStr[10];
strcpy(myStr, "LINUX");// Okay: 5
int myStrLen = strlen(myStr);// Okay: 5
int myStrCmp = strncmp(myStr, "LINUX", 5);// Okay: 0 (identical)
int myStrCmp = strncmp(myStr, "LINUX", 4);// Okay: 0 (identical)
int myStrCmp = strncmp(myStr, "LINUX", 6);// Okay: -1 (myStr < "LINUX")
int myStrCmp = strncmp(myStr, "LINUX", 3);// Okay: 1 (myStr > "LINUX")
int myStrCmp = strncmp(myStr, "LINUX", 0);// Okay: 0 (0 characters compared)
int myStrCmp = strncmp(myStr, "LINUX", -1);// Okay: 0 (0 characters compared)
int myStrCmp = strncmp(myStr, "LINUX", -2);// Okay: 0 (0 characters compared)
```

Tasks

Download the starting code. It contains all the needed to compile and finish the tasks. You may also download **sample compilation**. You can find an executable compilation for your operating system and architecture in our sample binaries on your course machine. Note that the binaries: lab07_linux and lab07_mac are intended for optional tests. They will not be used for grading.

There are 4 required tasks and 1 optional (expandable task). All tasks start with:

```
// TODO: Task X
```

It should be obvious where each task ends.

Each task comes with detailed documentation that tells you what to do. Read them carefully. For convenience, the documentation is duplicated here.

Task 1

- file: dictionary.dpp
- function: dictionary_entry_init

Task 1 implement the functions:

Remember that K-V-L (non-dictionary) data members are not initialized by default. You need to initialize them, or else you may get garbage values.

The function should create a dictionary entry with the given key and value. It should:

- There is no dict DEFINITION_LENGTH or DEFINITION to exist in dictionary.dpp (check yourself).
- Initialize word and definition to the empty C-string (C-string with a single '\0' character).
- Initialize next to NULL.
- Initialize definition to a C-string that is a copy of definition, truncated to the first MAXIMUM_DEFINITION_LENGTH (256) characters, including the null terminator.

You are recommended to use the C-string functions introduced in the lab webpage.

Task 2

- file: dictionary.dpp
- function: dictionary_entry_copy

Task 2 implement the function:

A copy-ctor/copy-assignment-operator-like, but you will see you can reuse the function in later tasks.

dictionary is a reference to a variable storing a pointer to the first entry of a dictionary. The variable stores NULL(0)if it the dictionary is empty. The function should create a dictionary entry with the same key and value as the given entry, and return a pointer to the new entry. It should:

```
// This function will be used in dictionary_add_entry()
dictionary_entry *new_entry = dictionary_entry_copy(dictionary, entry);// This is used as a dictionary entry.
// Now dictionary points to the first entry of the dictionary.
// Now new_entry points to the first entry of the dictionary.

dictionary_entry *new_entry2 = dictionary_entry_copy(dictionary, entry);// This is used as a dictionary entry.
// Now dictionary points to the first entry of the dictionary.
// Now new_entry points to the second entry of the dictionary.
```

- There is no dict DEFINITION_LENGTH or DEFINITION to exist in dictionary.dpp (check yourself).
- Initialize word and definition to the empty C-string (C-string with a single '\0' character).
- Initialize next to NULL.
- Initialize definition to a C-string that is a copy of definition, truncated to the first MAXIMUM_DEFINITION_LENGTH (256) characters, including the null terminator.

You are recommended to use the C-string functions introduced in the lab webpage.

Task 3

- file: dictionary.dpp
- function: dictionary_add_entry

Task 3 implement the function:

Using copy-ctor/copy-assignment-operator-like, but you will see you can reuse the function in later tasks.

dictionary is a reference to a variable storing a pointer to the first entry of a dictionary. The variable stores NULL(0)if it the dictionary is empty. The function should add a dictionary entry with the given key and value to the dictionary. It should:

```
// This function will be used in dictionary_add_entry()
dictionary_add_entry(dictionary, new_entry);// Insert an entry to the dictionary at the start.
// Now dictionary points to the first entry of the dictionary.
// Now new_entry points to the first entry of the dictionary.

dictionary_add_entry(dictionary, new_entry);// Insert an entry to the dictionary at the start.
// Now dictionary points to the first entry of the dictionary.
// Now new_entry points to the second entry of the dictionary.

dictionary_add_entry(dictionary, "new");// Remove the entry with the word "new" from the dictionary, if
// Now dictionary points to the first entry of the dictionary.
```

- There is no need to "delete" stale dictionary_entry entry because the generic entry function
- To implement the functions, you may find the dictionary_find_first_entry and dictionary_find_next_entry useful. Note, since a set can appear at most once in a dictionary, the dictionary_find_next_entry function will not be useful for this task.

If you find an entry with the same word as the entry to be inserted, the target is the first such entry. Take care of the case where the entry to be removed is the first entry in the dictionary. Return 1(true) if a dictionary was found and removed an entry. Remember to delete the entry so that it does not cause a memory leak.

Otherwise, return 0(false) to indicate you have not found an entry with the specified word to delete.

You are recommended to use the C-string functions introduced in the lab webpage.

Task 3

- file: dictionary.dpp
- function: dictionary_add_entry

Task 3 implement the function:

Use Task 2 to create a dictionary entry of dictionary.

For replacing the old entry, there is a very easy way to ensure the old entry is removed...

Only adding the new entry, and it is inserted at the start, you do not need to iterate through the dictionary to find its old to insert the new entry.

As a reference, the solution code for this task is very simple.

Task 4

- file: dictionary.dpp
- function: dictionary_clear

Task 4: implement the function.

See Task 2 for a detailed explanation of dictionary.

Ensure there are no memory leaks.

There is also a very easy way to implement this function... As a reference, the solution code for this task is very simple, even simpler than task 3.

Task 5

- file: dictionary.dpp
- function: dictionary_print

Task 5: implement the function.

See Task 2 for a detailed explanation of dictionary.

Print the dictionary in the following format: Print each entry in the dictionary in order, using: [word]: [definition] (note the space after the colon). Print a newline after each entry. Then, finally, print a summary line: dictionary of words: [number of words] (note the dictionary), and then a newline. If there is no dictionary, the provided testcase outputs nothing over this description.

Otherwise, return 0(false) to indicate you have not found an entry with the specified word to delete.

You are recommended to use the C-string functions introduced in the lab webpage.

Task 7 (optional)

- file: dictionary.dpp
- function: dictionary_search

Task 7 (optional): implement this function. This task is optional, so complete it.

Feel free to use any working function you know. Or, if you want to, feel free to simple working algorithm (but not practical in the real world).

1. Create a new entry dictionary.
2. Find the lexicographically largest word in the provided dictionary.
3. Copy the corresponding dictionary entry and insert it to the new dictionary at the start.
4. Find the second lexicographically largest word in the provided dictionary.
5. Copy the corresponding dictionary entry and insert it to the new dictionary at the start.
6. Repeat until you have copied the number of entries specified.

Ensure there are no memory leaks. Ensure the original dictionary is unchanged.

You are recommended to use the C-string functions introduced in the lab webpage.

Testcases for this optional task are named testcase-task7.txt and testcase-task7-MT.txt.

Sample Input/Output

You may find these as extra testcases.

The input values are highlighted.

Task 1

```
Welcome to dictionary editor!
----------------------------------------
1. Quit
2. Copy a new dictionary entry (but does not edit the dictionary)
3. Delete a dictionary entry
4. Add a dictionary entry
5. Clear the dictionary
6. Print the dictionary
7. Query the dictionary by partial word match (ignores definition)
8. Sort the dictionary
Option: 2
Word: apple
Definition: an fruit
Word added
----------------------------------------
1. Quit
2. Copy a new dictionary entry (but does not edit the dictionary)
3. Delete a dictionary entry
4. Add a dictionary entry
5. Clear the dictionary
6. Print the dictionary
7. Query the dictionary by partial word match (ignores definition)
8. Sort the dictionary
Option: 2
Word: bear
Definition: an animal
Word added
----------------------------------------
1. Quit
2. Copy a new dictionary entry (but does not edit the dictionary)
3. Delete a dictionary entry
4. Add a dictionary entry
5. Clear the dictionary
6. Print the dictionary
7. Query the dictionary by partial word match (ignores definition)
8. Sort the dictionary
Option: 2
Word: cat
Definition: an animal
Word added
----------------------------------------
1. Quit
2. Copy a new dictionary entry (but does not edit the dictionary)
3. Delete a dictionary entry
4. Add a dictionary entry
5. Clear the dictionary
6. Print the dictionary
7. Query the dictionary by partial word match (ignores definition)
8. Sort the dictionary
Option: 2
Word: dog
Definition: an animal
Word added
----------------------------------------
1. Quit
2. Copy a new dictionary entry (but does not edit the dictionary)
3. Delete a dictionary entry
4. Add a dictionary entry
5. Clear the dictionary
6. Print the dictionary
7. Query the dictionary by partial word match (ignores definition)
8. Sort the dictionary
Option: 6
apple: an fruit
bear: an animal
cat: an animal
dog: an animal
dictionary of words: 4
----------------------------------------
1. Quit
2. Copy a new dictionary entry (but does not edit the dictionary)
3. Delete a dictionary entry
4. Add a dictionary entry
5. Clear the dictionary
6. Print the dictionary
7. Query the dictionary by partial word match (ignores definition)
8. Sort the dictionary
Option: 7
Partial word: ar
Matching dictionary with word: ar
no word: apple definition
bear: an animal
cat: an animal
dog: an animal
Number of words: 3
----------------------------------------
1. Quit
2. Copy a new dictionary entry (but does not edit the dictionary)
3. Delete a dictionary entry
4. Add a dictionary entry
5. Clear the dictionary
6. Print the dictionary
7. Query the dictionary by partial word match (ignores definition)
8. Sort the dictionary
Option: 8
Word: cat
Word deleted
----------------------------------------
1. Quit
2. Copy a new dictionary entry (but does not edit the dictionary)
3. Delete a dictionary entry
4. Add a dictionary entry
5. Clear the dictionary
6. Print the dictionary
7. Query the dictionary by partial word match (ignores definition)
8. Sort the dictionary
Option: 3
Word: bear
Word deleted
----------------------------------------
1. Quit
2. Copy a new dictionary entry (but does not edit the dictionary)
3. Delete a dictionary entry
4. Add a dictionary entry
5. Clear the dictionary
6. Print the dictionary
7. Query the dictionary by partial word match (ignores definition)
8. Sort the dictionary
Option: 7
Partial word: ar
Matching dictionary with word: ar
no word: apple definition
dog: an animal
Number of words: 2
----------------------------------------
1. Quit
2. Copy a new dictionary entry (but does not edit the dictionary)
3. Delete a dictionary entry
4. Add a dictionary entry
5. Clear the dictionary
6. Print the dictionary
7. Query the dictionary by partial word match (ignores definition)
8. Sort the dictionary
Option: 5
Dictionary cleared
----------------------------------------
1. Quit
2. Copy a new dictionary entry (but does not edit the dictionary)
3. Delete a dictionary entry
4. Add a dictionary entry
5. Clear the dictionary
6. Print the dictionary
7. Query the dictionary by partial word match (ignores definition)
8. Sort the dictionary
Option: 6
dictionary empty
----------------------------------------
1. Quit
2. Copy a new dictionary entry (but does not edit the dictionary)
3. Delete a dictionary entry
4. Add a dictionary entry
5. Clear the dictionary
6. Print the dictionary
7. Query the dictionary by partial word match (ignores definition)
8. Sort the dictionary
Option: 7
Partial word: ar
Querying dictionary with word: ar
no word: apple definition
dog: an animal
Number of words found: 1
----------------------------------------
1. Quit
2. Copy a new dictionary entry (but does not edit the dictionary)
3. Delete a dictionary entry
4. Add a dictionary entry
5. Clear the dictionary
6. Print the dictionary
7. Query the dictionary by partial word match (ignores definition)
8. Sort the dictionary
Option: 3
Word: dog
Word deleted
----------------------------------------
1. Quit
2. Copy a new dictionary entry (but does not edit the dictionary)
3. Delete a dictionary entry
4. Add a dictionary entry
5. Clear the dictionary
6. Print the dictionary
7. Query the dictionary by partial word match (ignores definition)
8. Sort the dictionary
Option: 7
Partial word: ar
Querying dictionary with word: ar
no word: apple definition
Number of words found: 0
----------------------------------------
1. Quit
2. Copy a new dictionary entry (but does not edit the dictionary)
3. Delete a dictionary entry
4. Add a dictionary entry
5. Clear the dictionary
6. Print the dictionary
7. Query the dictionary by partial word match (ignores definition)
8. Sort the dictionary
Option: 8
Word: apple
Word deleted
----------------------------------------
1. Quit
2. Copy a new dictionary entry (but does not edit the dictionary)
3. Delete a dictionary entry
4. Add a dictionary entry
5. Clear the dictionary
6. Print the dictionary
7. Query the dictionary by partial word match (ignores definition)
8. Sort the dictionary
Option: 6
dictionary empty
----------------------------------------
1. Quit
2. Copy a new dictionary entry (but does not edit the dictionary)
3. Delete a dictionary entry
4. Add a dictionary entry
5. Clear the dictionary
6. Print the dictionary
7. Query the dictionary by partial word match (ignores definition)
8. Sort the dictionary
Option: 2
Word: apple
Definition: an fruit
Word added
----------------------------------------
1. Quit
2. Copy a new dictionary entry (but does not edit the dictionary)
3. Delete a dictionary entry
4. Add a dictionary entry
5. Clear the dictionary
6. Print the dictionary
7. Query the dictionary by partial word match (ignores definition)
8. Sort the dictionary
Option: 2
Word: bear
Definition: an animal
Word added
----------------------------------------
1. Quit
2. Copy a new dictionary entry (but does not edit the dictionary)
3. Delete a dictionary entry
4. Add a dictionary entry
5. Clear the dictionary
6. Print the dictionary
7. Query the dictionary by partial word match (ignores definition)
8. Sort the dictionary
Option: 2
Word: cat
Definition: an animal
Word added
----------------------------------------
1. Quit
2. Copy a new dictionary entry (but does not edit the dictionary)
3. Delete a dictionary entry
4. Add a dictionary entry
5. Clear the dictionary
6. Print the dictionary
7. Query the dictionary by partial word match (ignores definition)
8. Sort the dictionary
Option: 2
Word: dog
Definition: an animal
Word added
----------------------------------------
1. Quit
2. Copy a new dictionary entry (but does not edit the dictionary)
3. Delete a dictionary entry
4. Add a dictionary entry
5. Clear the dictionary
6. Print the dictionary
7. Query the dictionary by partial word match (ignores definition)
8. Sort the dictionary
Option: 6
apple: an fruit
bear: an animal
cat: an animal
dog: an animal
dictionary of words: 4
----------------------------------------
1. Quit
2. Copy a new dictionary entry (but does not edit the dictionary)
3. Delete a dictionary entry
4. Add a dictionary entry
5. Clear the dictionary
6. Print the dictionary
7. Query the dictionary by partial word match (ignores definition)
8. Sort the dictionary
Option: 7
Partial word: ar
Matching dictionary with word: ar
no word: apple definition
bear: an animal
cat: an animal
dog: an animal
Number of words: 3
----------------------------------------
1. Quit
2. Copy a new dictionary entry (but does not edit the dictionary)
3. Delete a dictionary entry
4. Add a dictionary entry
5. Clear the dictionary
6. Print the dictionary
7. Query the dictionary by partial word match (ignores definition)
8. Sort the dictionary
Option: 8
Word: bear
Word deleted
----------------------------------------
1. Quit
2. Copy a new dictionary entry (but does not edit the dictionary)
3. Delete a dictionary entry
4. Add a dictionary entry
5. Clear the dictionary
6. Print the dictionary
7. Query the dictionary by partial word match (ignores definition)
8. Sort the dictionary
Option: 3
Word: cat
Word deleted
----------------------------------------
1. Quit
2. Copy a new dictionary entry (but does not edit the dictionary)
3. Delete a dictionary entry
4. Add a dictionary entry
5. Clear the dictionary
6. Print the dictionary
7. Query the dictionary by partial word match (ignores definition)
8. Sort the dictionary
Option: 7
Partial word: ar
Querying dictionary with word: ar
no word: apple definition
dog: an animal
Number of words found: 2
----------------------------------------
1. Quit
2. Copy a new dictionary entry (but does not edit the dictionary)
3. Delete a dictionary entry
4. Add a dictionary entry
5. Clear the dictionary
6. Print the dictionary
7. Query the dictionary by partial word match (ignores definition)
8. Sort the dictionary
Option: 8
Word: dog
Word deleted
----------------------------------------
1. Quit
2. Copy a new dictionary entry (but does not edit the dictionary)
3. Delete a dictionary entry
4. Add a dictionary entry
5. Clear the dictionary
6. Print the dictionary
7. Query the dictionary by partial word match (ignores definition)
8. Sort the dictionary
Option: 7
Partial word: ar
Querying dictionary with word: ar
no word: apple definition
Number of words found: 0
----------------------------------------
1. Quit
2. Copy a new dictionary entry (but does not edit the dictionary)
3. Delete a dictionary entry
4. Add a dictionary entry
5. Clear the dictionary
6. Print the dictionary
7. Query the dictionary by partial word match (ignores definition)
8. Sort the dictionary
Option: 5
Dictionary cleared
----------------------------------------
1. Quit
2. Copy a new dictionary entry (but does not edit the dictionary)
3. Delete a dictionary entry
4. Add a dictionary entry
5. Clear the dictionary
6. Print the dictionary
7. Query the dictionary by partial word match (ignores definition)
8. Sort the dictionary
Option: 6
dictionary empty
----------------------------------------
1. Quit
2. Copy a new dictionary entry (but does not edit the dictionary)
3. Delete a dictionary entry
4. Add a dictionary entry
5. Clear the dictionary
6. Print the dictionary
7. Query the dictionary by partial word match (ignores definition)
8. Sort the dictionary
Option: 2
Word: apple
Definition: an fruit
Word added
----------------------------------------
1. Quit
2. Copy a new dictionary entry (but does not edit the dictionary)
3. Delete a dictionary entry
4. Add a dictionary entry
5. Clear the dictionary
6. Print the dictionary
7. Query the dictionary by partial word match (ignores definition)
8. Sort the dictionary
Option: 2
Word: bear
Definition: an animal
Word added
----------------------------------------
1. Quit
2. Copy a new dictionary entry (but does not edit the dictionary)
3. Delete a dictionary entry
4. Add a dictionary entry
5. Clear the dictionary
6. Print the dictionary
7. Query the dictionary by partial word match (ignores definition)
8. Sort the dictionary
Option: 2
Word: cat
Definition: an animal
Word added
----------------------------------------
1. Quit
2. Copy a new dictionary entry (but does not edit the dictionary)
3. Delete a dictionary entry
4. Add a dictionary entry
5. Clear the dictionary
6. Print the dictionary
7. Query the dictionary by partial word match (ignores definition)
8. Sort the dictionary
Option: 2
Word: dog
Definition: an animal
Word added
----------------------------------------
1. Quit
2. Copy a new dictionary entry (but does not edit the dictionary)
3. Delete a dictionary entry
4. Add a dictionary entry
5. Clear the dictionary
6. Print the dictionary
7. Query the dictionary by partial word match (ignores definition)
8. Sort the dictionary
Option: 6
apple: an fruit
bear: an animal
cat: an animal
dog: an animal
dictionary of words: 4
----------------------------------------
1. Quit
2. Copy a new dictionary entry (but does not edit the dictionary)
3. Delete a dictionary entry
4. Add a dictionary entry
5. Clear the dictionary
6. Print the dictionary
7. Query the dictionary by partial word match (ignores definition)
8. Sort the dictionary
Option: 7
Partial word: ar
Matching dictionary with word: ar
no word: apple definition
bear: an animal
cat: an animal
dog: an animal
Number of words: 3
----------------------------------------
1. Quit
2. Copy a new dictionary entry (but does not edit the dictionary)
3. Delete a dictionary entry
4. Add a dictionary entry
5. Clear the dictionary
6. Print the dictionary
7. Query the dictionary by partial word match (ignores definition)
8. Sort the dictionary
Option: 8
Word: apple
Word deleted
----------------------------------------
1. Quit
2. Copy a new dictionary entry (but does not edit the dictionary)
3. Delete a dictionary entry
4. Add a dictionary entry
5. Clear the dictionary
6. Print the dictionary
7. Query the dictionary by partial word match (ignores definition)
8. Sort the dictionary
Option: 3
Word: bear
```

```
6. Query the dictionary by partial word match (ignores definition)
7. Sort the dictionary
Option: 5
Printing dictionary
word: def
not word: not def
Number of words: 2
==========================================
0. Quit
1. Copy a new dictionary entry (but does not edit the dictionary)
2. Delete a dictionary entry
3. Add a dictionary entry
4. Clear the dictionary
5. Print the dictionary
6. Query the dictionary by partial word match (ignores definition)
7. Sort the dictionary
Option: 7
Dictionary sorted
==========================================
0. Quit
1. Copy a new dictionary entry (but does not edit the dictionary)
2. Delete a dictionary entry
3. Add a dictionary entry
4. Clear the dictionary
5. Print the dictionary
6. Query the dictionary by partial word match (ignores definition)
7. Sort the dictionary
Option: 5
Printing dictionary
not word: not def
word: def
Number of words: 2
==========================================
0. Quit
1. Copy a new dictionary entry (but does not edit the dictionary)
2. Delete a dictionary entry
3. Add a dictionary entry
4. Clear the dictionary
5. Print the dictionary
6. Query the dictionary by partial word match (ignores definition)
7. Sort the dictionary
Option: 0
Quitting!
```

Checking Before Submission

ZINC testing instructions

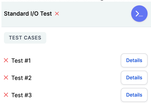
You should upload your modified **dictionary.cpp** to ZINC. You don't need to zip the cpp file if you only upload a single C++ file.

- ZINC submission platform

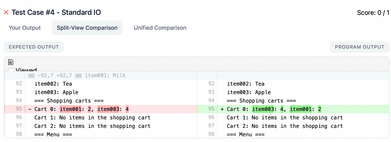
Please compare your outputs with the desired outputs in ZINC:

- You should see the following if the test cases are correct:

- You should see the following if the test cases are wrong:



- For the wrong cases, you can click the "Details" button, and then check the side-by-side comparsion
- You may see an error report like this (for illustration, not related to the actual test cases):



  - The left side is the expected output
  - The right side is your program output
  - The differences are highlighted in red and green

Local testing instructions

- **DO NOT** use **cerr** because ZINC cannot grade cerr output. You can check this problem by following the checking procedures below.
- **DO NOT** return a non-zero return code (e.g., return -1;) in the main function. ZINC will treat your program as "crash" and won't continue the grading.

Please compare your outputs with the desired outputs:

- In VSCode, please open the terminal by one of the following two ways:
  - Press the key Ctrl+Shift+` in Windows/Linux, or Control+Shift+Backquote in Mac
  - The shortcut keys are for VSCode
  - From the menubar, select Terminal > New Terminal

  And then run this code to compile your code:

  ```
  g++ -o main -std=c++11 -g -Wall -Wextra -Wpedantic -fsanitize=address,leak,undefined dictionary.cpp main.cpp
  ```

  The sanitizer option `-fsanitize=address,leak,undefined` is aimed at checking memory leak of your code. If you encounter some errors relevant with the sanitizers, you may use the command without sanitizer option:

  ```
  g++ -o main -std=c++11 -g -Wall -Wextra -Wpedantic dictionary.cpp main.cpp
  ```

  Note if you use this compilation command, you cannot check the memory leak for your source code.
  An alternative method is using a Makefile. A Makefile allows you to associate names with commands so that you do not need to type the above command over and over again. It also allows faster compilation times by selectively recompiling files that you have modified since last compilation. Makefile will be taught in COMP 2012 labs. For now, we have provided a Makefile in the starting code for you. You do not need to submit it.
  - To compile the executable:

    ```
    make
    ```

  - To use Makefile to compile an executable that also detects bad memory accesses, memory leaks, and undefined behaviors:

    ```
    make debug
    ```

    This command is not available on Windows.
  - To remove intermediate files created by compilation:

    ```
    make clean
    ```

  - To remove all files created by compilation, including the executables:

    ```
    make distclean
    ```

- You need to redirect the input of executable program to the txt test case files we provided, and redirect the output to different txt files. Here are the commands:
  - In MacOS or Linux:

    ```
    ./main < testcase/input1.txt > myOutput1.txt
    ```

  - In Windows (PowerShell):

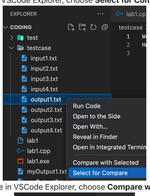
    ```
    Get-Content .\testcase\input1.txt | .\main.exe > myOutput1.txt
    ```

  - In Windows (CMD):

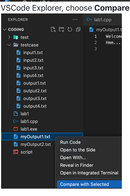
    ```
    .\main.exe < .\testcase\input1.txt > myOutput1.txt
    ```

  Please replace the filenames accordingly if you would like to check other output files.

- After that, you can compare your own output text files with the desired outputs one by one. You can compare any two files by VSCode:
  - Right click the first file you want to compare in VSCode Explorer, choose **Select for Compare**



  - Right click the second file you want to compare in VSCode Explorer, choose **Compare with Selected**



Grading Scheme

Each lab carries 3 points in total.

- A student gets 1 point for attendance if the student shows up in the first 10 minutes of the lab session.
- A student gets a maximum of 1 point for a successful ZINC submission (**dictionary.cpp**, the filename is case-sensitive) that passes ALL the seen and unseen test cases.
  The ZINC submission deadline is 10 minutes after your lab class. For example, if the lab end time is 12:50PM, the ZINC submission deadline is 01:00PM (i.e., 10 minutes after 12:50PM) on the same day.
- Passing the given ZINC test cases may not guarantee you to get full marks in ZINC. After the zinc submission due date, there will be additional test cases which are similar to the given ones for grading purpose.
- A student gets a maximum of 1 point for the Q&A session with the TA if he/she gives the correct answer to TA's question.
  - The list of selected students will only be shown during the lab session.
  - If the student is not selected for the Q&A, then the ZINC submission will count for 2 points.
    - Some examples:
      - If a student attended the lab, was not selected in Q and A, and passed all test cases in ZINC.
        *1 (from attendance) + 1 (from ZINC) * 2 = 3 marks*
      - If a student attended the lab, was not selected in Q and A, and passed half (50%) of the test cases in ZINC.
        *1 (from attendance) + 0.5 (from ZINC) * 2 = 2 marks*
      - If a student attended the lab, was not selected in Q and A, and failed all test cases in ZINC.
        *1 (from attendance) + 0 (from ZINC) * 2 = 1 mark*
- The filename is **dictionary.cpp**, and submit it to ZINC. **DO NOT** submit a file with a wrong filename.
- You don't need to zip the file if you only submit one file to ZINC. Filename checking is enabled in ZINC. If your filename is wrong, you should see something like this:

If a student is unable to finish the lab and demonstrate to the TA by the end of the lab, they can only get the attendance point.