HO CHI MINH UNIVERSITY OF SCIENCE
FACULTY OF INFORMATION TECHNOLOGY
SOFTWARE ENGINEERING DEPARTMENT
ADVANCED PROGRAM IN COMPUTER SCIENCE
COURSE: **DATA STRUCTURE**
LECTURER: Dr. ĐINH BÁ TIẾN

WEEK 01

HASH TABLE

- TRƯƠNG PHƯỚC LỘC
- HỒ TUẤN THANH

HCMC, 2016

1 Hash Table

- Hash table - http://www.algolist.net/Data_structures/Hash_table
- Simple hash table - http://www.algolist.net/Data_structures/Hash_table/Simple_example
- Examples:
  - http://www.algolist.net/Data_structures/Hash_table/Chaining
  - http://www.algolist.net/Data_structures/Hash_table/Open_addressing
  - http://www.algolist.net/Data_structures/Hash_table/Dynamic_resizing

2 Exercise

Your program will count the number of occurrences of words in a text file.

Example:

Input:

```
to
be
or
not
to
be
that
is
the
question
whether
```

Output (order is not important):

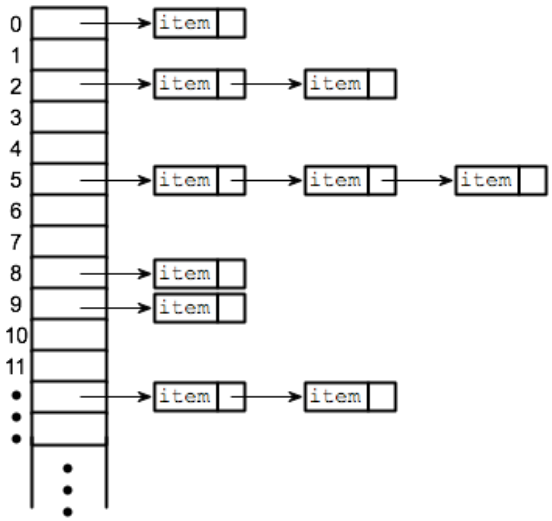
```
to 2
be 2
or 1
not 1
that 1
is 1
the 1
question 1
```

```
whether 1
```

3 Guide

Using the supporting files to write the program.

1. Implementing a hash table
    - Number of buckets: 128 (it should be defined as an parameter to initialize a hash table)
    - Type of key (word): char *
    - Type of value (number of occurrences): int
    - Hash function:



Ideally, the hash function will assign each key to a unique bucket. But the number of words is unknown, the number of buckets is limited. Therefore, collisions are practically unavoidable when hashing a random subset of a large set of possible keys.

The linked lists is used to solve the collisions. Each bucket is a pointer to a node (head) of the list of nodes which share the same key (NULL if there no key in this bucket). Each node consists: a key, a value and a pointer to the next node (NULL in case of the last node).

Implementing these operations:

- Initialize a hash table (char * to int) with the number of buckets (default 128)
  - Hash the key to an index
  - Find the node consists the key in the linked lists
  - Return the search result (true or false)
- Get value:
  - Hash the key to an index
  - Find the node consists the key in the linked lists
  - Return the value of the found node
- Set value:
  - Hash the key to an index
  - Find the node consists the key in the linked lists
    - Found: set the new value of the found node
    - Not found: create the new node for the key and its value
- Print: Print all the keys and its value as key => value

```
to 2
be 1
```

- Destroy a hash table: Free allocated memory

2. Using a hash table to count the number of occurrences

   For each input word:

   - Look up the word in the hash table;
   - If it isn't there, add it to the hash table with a value of 1;
   - If it is there, add 1 to its value.

3. Free memory
   - All nodes in the linked lists;
   - Hash table array;
   - String keys;

- …

Reference: http://courses.cms.caltech.edu/cs11/material/c/mike/lab7/lab7.html