Data structure and Programming II

TP11: Hash Table

- Objective:
  - To practice implementing hash table data structure in program to increase search performance
  - Apply C++ programming
- Individual work
- Submit to Moodle (do not zip)
  - *A pdf report (cover, exercise and solution with screenshot)*
  - *Source codes*

Deadline:
6 days

---

1. (**No collision**) Consider a hash table data structure that has a simple hash function as "key mod 20". We would like to insert these numbers 50, 700, 76, 85, 92, 73, 101, respectively into this hash table. Draw the hash table to see how your data is stored.

2. (**No collision**) Create a hash table to store 4 integer numbers. This hash table uses a hash function to compute the index using the operation data % size of array, assume that the size of array is 7. Write 4 functions to
   - Insert a number to the hash table.
   - Display all data in hash table. Each data should be displayed with its index
   - Search whether a number is contained in the hash table. If the number is contained in the hash table, then display a message "It is contained in position ***positionNumber***" and return true. Otherwise, display "Not found" and return false.
   - Remove a number from the hash table. If the number is contained in the hash table, then remove it by setting a default value (e.g: for array of integer, the default value for each element can be 0). Finally, display a message "removed success. Otherwise (if not contained), display "failed to delete as it does not exist".

In main program, add the numbers of 2, 4, 7, 10. Search for the number 7 by calling the search function. Then remove the number 10 using the remove function. Finally, display all data in the hash table.

Hint: The hash value returning from the hash function is the index of the number in the hash table.

2 mod 7 is **2** => index of number 2 is 2

4 mod 7 is **4** => index of number 4 is 4

7 mod 7 is **0** => index of number 7 is 0

10 mod 7 is **3** => index of number 10 is 3

Hash Table	
Index	Number
0	7
1	
2	2
3	10
4	4
5	
6	

```
int main(){
    insert(2);
    insert(4);
    insert(7);
    insert(10);
    display();
    cout<<"\n";
}
```

```
Index: 0, Data: 7
Index: 1, Data: 0
Index: 2, Data: 2
Index: 3, Data: 10
Index: 4, Data: 4
Index: 5, Data: 0
Index: 6, Data: 0
```

After using insert function

```
int main(){
    insert(2);
    insert(4);
    insert(7);
    insert(10);
    search(7);
    remove(10);
    display();
    cout<<"\n";
}
```

```
It is contained in position/index 0

removed success

Index: 0, Data: 7
Index: 1, Data: 0
Index: 2, Data: 2
Index: 3, Data: 0
Index: 4, Data: 4
Index: 5, Data: 0
Index: 6, Data: 0
```

After using search and remove

An example of Ex2's output

3. (**No collision**) Create a hash table with the size of 10 to store names of people. This hash table uses a hash function which returns the index in which the name should be inserted into. The hash function computes the total summation of ASCII value in the string (sum each character's ASCII value together) and modulo with 10. Write 3 functions to:
   - Insert a string to the hash table
   - Search whether a string is contained in the hash table. If the string is contained in the hash table, then display a message "It is contained in position/index ***positionNumber***". Otherwise, display "Not found".
   - Remove the string from the hash table. If the string is contained in the hash table, then remove it by setting to a default value (e.g: for array of string, the default value can be an empty string). Finally display a message "removed success". Otherwise (if not contained), display "failed to delete as it does not exist".

In main program, add a string "Bob" to the hash table. Then ask a user for 3 other names and add to the hash table. Search for the string "Bob" by calling search function. Then remove string "Bob" using the remove function. Finally, display all data in the hash table.

Hint: The hash value returning from the hash function is the index of the string in the hash table. See example below for the string "Bob":

Hash Table	
Index	String
0	
1	
2	
3	
4	
5	Bob
6	
7	
8	
9	

Add "Bob" to hash table

=> (66+111+98) mod 10 is 5 => index of Bob is 5

Remark: Apply chaining to solve the collision problem.

4. Create a hash table to store key-value data for 15 students.
A student has **ID**, **name**, **sex**, **phone** and **major**.

- A student's ID will be taken as a key which is used as a parameter for the hash function. The value to be inserted to an array slot at the position defined by the hash function is a student info of that ID.
- To avoid collision, you have to apply collision resolution technique called chaining (implement hash table as an array of linked list).
- The hash function of this hash table is to generate the student ID into an index of the array. Let assume that the student ID is the same format as ITC' engineer student ID (e.g: e20110015)

To test this program, enter information of 3 students (your friends) then

a. Insert their information to the hash table.
b. Finally, display all data in hash table.
c. Study the case that has duplicate data in order to verify that your program is still working well since we apply collision resolution technique.