Lab 11: Hashing

In this lab, we will implement hashing ADT. For resolving collisions, you may want to use open addressing with linear probing (i.e. F(i) = i).

1. Input

Your input file consists of three lines.

* In the first line, your hash table size is given.
* In the second line, you obtain all the data that should be inserted into the hash table. Obtain a list of numbers from the second line, and execute an insertion operation for each number in order. If a collision happens, print a meesage to notify. Duplicated insertion query will be rejected. Don't notify when insertion succeeds.
* In the third line, the numbers are given for checking whether each number is in the hash table or not. For each number, print the message about the availability.

Input.txt

|  |
| --- |
| 30  3 5 35 2 7 18 19 22 5 100 26 8 4 16  5 27 45 67 2 |

2. Hashing ADT

(1) Data Specification for the hash table

typedef int ElementType;

struct HashTable{

int TableSize;

ElementType\* TheLists;

};

(2) functions

void Insert (ElementType Key, struct HashTable \*H)

* print an error message when a duplicated key is insert (request will be rejected)
* print a message when a collision occurs
* do NOT print any message when insertion succeeds
* collision resolution methods are given at the top of the lecture note

int find(struct HashTable \*H, ElementType value)

//will return non zero value when succeed to find. If not, return 0.

3. Program description

* name : p11.c
* input : a list of operations in a file (an input file name is given as a command line argument. See the example in “1. input” on the first page)
* output : the corresponding result in the standard output

Submit to the course gitlab your source code. ( ~2020/6/4 23:59 )