

Lab 15 (6/9) Closed Hashing

Due : ~2022.06.14.(Tue) 23:59, Late Submission : ~2022.06.15.(Wed) 23:59

- **Closed Hashing Implementation - insert, delete, find, print.**
- 아래의 사진과 같이 [input].txt 파일을 입력받아 **Closed Hashing** 기능을 수행하고 결과값을 [output].txt 파일에 출력하여 저장한다.
- **Implement the function that creates and edits a closed Hash by receiving the [input].txt file as input commands as below, and the output result is stored in the [output].txt file.**

```

linear
11
i 1
i 11
i 4
i 15
i 22
f 64
i 22
i 9
i 18
i 77
i 16
d 4
d 18
d 85
p

insert 1 into address 1
insert 11 into address 0
insert 4 into address 4
insert 15 into address 5
insert 22 into address 2
64 is not in the table
insertion error : 22 already exists at address 2
insert 9 into address 9
insert 18 into address 7
insert 77 into address 3
insert 16 into address 6
delete 4
delete 18
deletion error : 85 is not in the table
11 1 22 77 0 15 16 0 0 9 0

```

- **<input> :**
각 line마다 **Command** 가 주어짐.(Collision Solution type, size, insert, find, print)
Commands are given line by line (Collision Solution type, size, insert, find, print)
- **<output> :**
각 **Command**에 맞는 **result** 출력.
Appropriate output messages for each command.

<collision Solution type & Hash Table size Setting>

- 위 사진에서처럼 첫째줄에 “**linear**” 혹은 “**quadratic**”로 입력됩니다.
- 위 사진에서처럼 둘째줄에 **100** 이하의 자연수로 입력됩니다.
- **“Linear” or “Quadratic”** in the first line as above.
- **Natural number less or equal to 100** in the second line as above.

<Insert>

- 위의 사진에서처럼 “**i [insert할 key값]**”로 표현합니다.
- insertion 성공시 : “**insert key값 into address entry인덱스**”
중복된 key값 insert시 : “**insertion error : key값 already exists at address entry인덱스**”
HashTable이 다 차있는 경우 : “**insertion error : table is full**”
- “**i [key value to be inserted]**”
- if insertion succeeds : “**insert keyValue into address entryIndex**”

if inserting duplicate key value : "insertion error : keyValue already exists at address entryIndex"

if HashTable is full : "insertion error : table is full"

<Find>

- 위의 사진처럼 "f [search 할 key값]"로 표현합니다.
- key 값을 찾았을 경우 출력 형식 : "key값 is in the table"
- key 값이 존재하지 않는 경우 출력 형식 : "key값 is not in the table"
- "f [key value to be searched]"
- if the key value is found : "keyValue is in the table"
- if the key value does not exist : "keyValue is not in the table"

<Delete>

- 위의 사진처럼 "d [delete 할 key값]"로 표현합니다.
- delete 성공시 출력 형식은 "delete key값"입니다.
- key 값이 존재하지 않는 경우 "deletion error : key값 is not in the table"입니다.
- "d [key value to be deleted]".
- if deletion succeeds : "delete keyValue".
- if the key value does not exist : "deletion error : keyValue is not in the table".

<Print>

- 위 사진처럼 "p"로 표현합니다.
- "p"의 경우, Hash Table의 index 순서대로 Hash Table 안에 있는 Key 값을 출력하시면 됩니다.
- 비어있는 Hash Table Entry의 경우 위 사진처럼 0으로 출력하시면 됩니다.
- "p"
- In the case of "p", all the key values in the Hash Table should be printed in order of Index.
- In the case of empty Entry, it should be printed as 0 as an example above.

<Hash Function & Collision Solution>

- $h(\text{value}) = \text{value \% Size}$, $i = \text{iteration \#}$
- $hi = (h(\text{value}) + i) \% \text{Size}$ (for linear), $hi = (h(\text{value}) + i*i) \% \text{Size}$ (for Quadratic)

<Structure & Function Format>

Structure

```
typedef int ElementType;
typedef ElementType List;
typedef struct HashTbl* HashTable;
typedef struct HashTbl{
    int TableSize;
    List *TheLists;
}HashTbl;
```

Function

```
HashTable createTable(int TableSize);
void Insert(HashTable H, ElementType Key, int solution);
void Delete(HashTable H, ElementType Key, int solution);
int Find(HashTable H, ElementType Key, int solution);
void printTable(HashTable H);
void deleteTable(HashTable H);
```

- 위 사진과 같은 Struct 구조체를 사용하셔야 합니다.
- 위 사진과 같은 함수들을 형식에 맞게 구현해주시면 됩니다.
- Struct format above should be used for implementation
- Functions should be implemented in appropriate format as above.

<File Name Format>

- [StudentID].c ex) 20XXXXXXXXXX.c

<Execution>

- gcc 20XXXXXXXXXX.c -o 20XXXXXXXXXX
 - ./20XXXXXXXXXX [input_file_name] [output_file_name]
 - !!! 꼭 제공되는 testcase로 실행시켜보시기 바랍니다!!!!,
 - !!! Run your solution code with the provided test case above and check whether it works properly !!!

<Issue>

- 코드 작성시 주석을 적어주시기 바랍니다. 주석이 없는경우 Cheating으로 간주될 수 있습니다.
 - 제공된 testcase는 채점 case에 포함됩니다. 모두 알맞게 나오는지 확인해보시기 바랍니다.
 - 파일 입출력은 `argv[]`를 사용하여 구현해주시기 바랍니다.
 - 제출 마감 시간 이전의 가장 최신 버전의 commit을 기준으로 채점할 예정입니다.
 - 제출 파일과, 폴더 naming 은 꼭 지정된 형식으로 해주셔야 합니다.
 - Please write down the detailed comments when writing the code. If there is no comment, it might be considered cheating.
 - Provided test case is included in the test case for grading. Please check to see if it makes a proper result.
 - Do not use a fixed file name when inputting and outputting files, but implement it using `argv[]` as in skeleton code.
 - Scoring will be based on the latest version of commit before the deadline.
 - The names of the .c file and directory should be named in proper format.
 - 출력시 꼭 정해진 형식에 맞게 출력해주셔야 합니다. 모든 공백은 띄어쓰기 한칸입니다.
-> 출력시 위 사진과 같이 커맨드에 알맞는 메시지 출력후 줄바꿈.
 - All the messages must be printed out according to the appropriate format as shown in the example above. All the spaces are “one space”.
-> `newline(\n)` after each output message.

<Directory Format>

- 아래와 같이 git 프로젝트 폴더에 "lab15" 폴더 생성후, "lab15" 폴더 안에 "20XXXXXXXXXX.c" 파일을 위치시키시면 됩니다.
 - After creating the "lab15" directory in the git-project-directory as below, place the "20XXXXXXXXXX.c" file in the "lab15" directory.

2022_CSE2010_20XXXXXXX/ (GitLab project directory)

— . . .

---lab13-1/

-----2020XXXXXX.c

— . . .

---lab13-2/

-----2020xxxxxx.c

Lab15

--lab15/
202

-----2020XXXXXX.0