Lab 14 & HW 14

Data Structure

Lab14 (due on the Lab Session)

1. Do p14_1.c

HW14 (due on the day before the next Lab Session)

1. Do p14_2.c

Evaluation criteria

Category	**Evaluation**	
p14_1	50	
p14_2	50	
Total	100	

- *Use GCC 4.8 version or GCC 5.4 version.*
- *No score will be given if the gcc version is different.*

Lab14 – Hashing

- You should finish p14_1 (open hashing) during the lab session and submit it on **portal site (assignment)** before you leave.
- For p14_2 (open addressing) you have to submit on **portal site (assignment)**
- code name: p14_1, p14_2
- No score, if the code names are wrong.
- No score, if it does not use FILE I/O
- Each code will be tested by 5 different input files.
- 10 score for each input, if you don't get the answer you get 0 score.

Lab14 - hashing

- **p14_1.c :** open hashing with division
- **p14_2.c :** open addressing with linear probing (F(i) =i)

- **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
  - collision resolution methods are given at the top of the lecture note
  - Print a error message when the hash table is full
  - Print message if inserted

- **Position find(struct HashTable *H, ElementType value)**
  - will return Node structure. If not, return NULL.

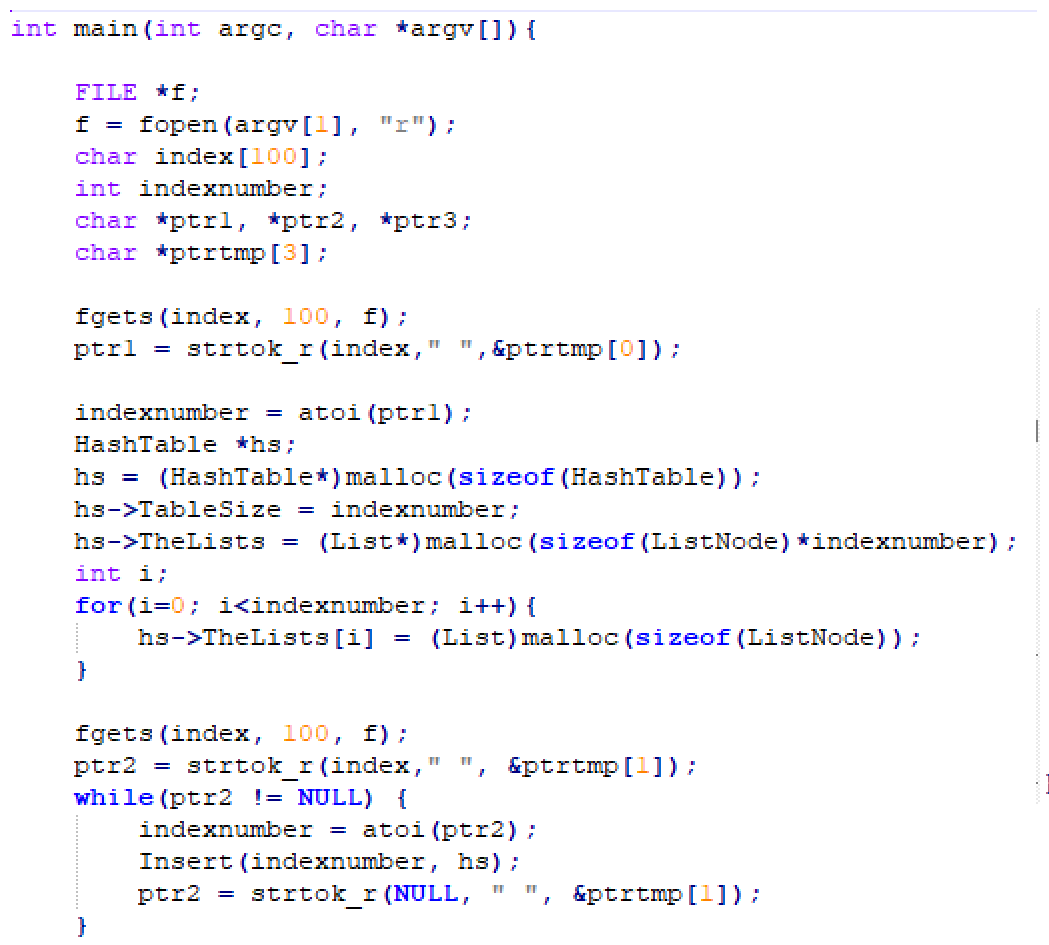
Lab14 – Hashing

- Structure

```
typedef struct ListNode {
            ElementType      Element;
            Position Next;
}ListNode;
```

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

Lab14 Hashing – open hashing



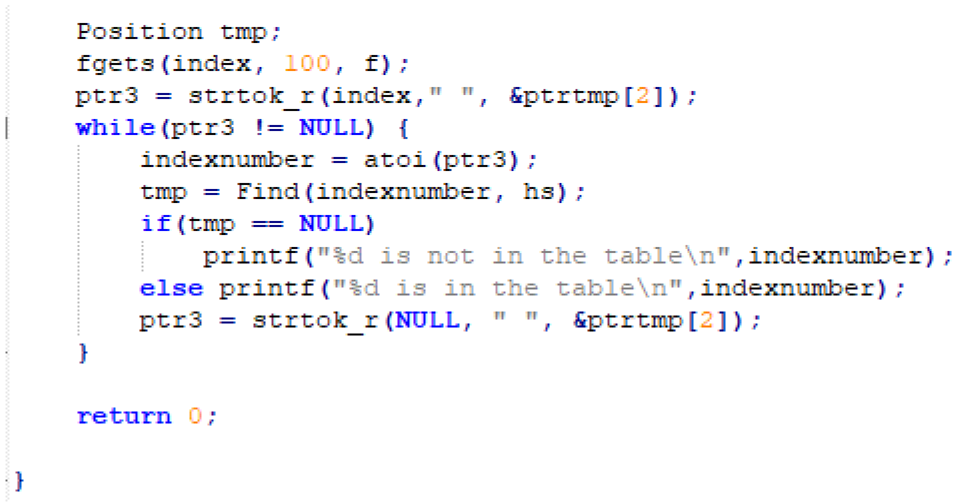
```c
int main(int argc, char *argv[]){

    FILE *f;
    f = fopen(argv[1], "r");
    char index[100];
    int indexnumber;
    char *ptr1, *ptr2, *ptr3;
    char *ptrtmp[3];

    fgets(index, 100, f);
    ptr1 = strtok_r(index," ",&ptrtmp[0]);

    indexnumber = atoi(ptr1);
    HashTable *hs;
    hs = (HashTable*)malloc(sizeof(HashTable));
    hs->TableSize = indexnumber;
    hs->TheLists = (List*)malloc(sizeof(ListNode)*indexnumber);
    int i;
    for(i=0; i<indexnumber; i++){
        hs->TheLists[i] = (List)malloc(sizeof(ListNode));
    }

    fgets(index, 100, f);
    ptr2 = strtok_r(index," ", &ptrtmp[1]);
    while(ptr2 != NULL) {
        indexnumber = atoi(ptr2);
        Insert(indexnumber, hs);
        ptr2 = strtok_r(NULL, " ", &ptrtmp[1]);
    }
```



```c
    Position tmp;
    fgets(index, 100, f);
    ptr3 = strtok_r(index," ", &ptrtmp[2]);
    while(ptr3 != NULL) {
        indexnumber = atoi(ptr3);
        tmp = Find(indexnumber, hs);
        if(tmp == NULL)
            printf("%d is not in the table\n",indexnumber);
        else printf("%d is in the table\n",indexnumber);
        ptr3 = strtok_r(NULL, " ", &ptrtmp[2]);
    }

    return 0;

}
```

Lab14. Hashing

- input file : Lab14_input1.txt

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

First line: your hash table size is given.

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 message to notify. Duplicated insertion query will be rejected. When hash bucket get dense, program should conduct rehashing.

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.

Lab14. Hashing – p14_1.c : open hashing

- Result

```
3 is inserted
5 is inserted
35 insertion collision has been occurred with number 5
35 is inserted
2 is inserted
7 is inserted
18 is inserted
19 is inserted
22 is inserted
5 is already in the table
100 is inserted
26 is inserted
8 is inserted
4 is inserted
16 is inserted
5 is in the table
27 is not in the table
45 is not in the table
67 is not in the table
2 is in the table
```

Lab14. Hashing – p14_2.c : open addressing

- Result

```
3 is inserted at address 3
5 is inserted at address 5
35 insertion collision has been occured with number 5
35 is inserted at address 6
2 is inserted at address 2
7 is inserted at address 7
18 is inserted at address 18
19 is inserted at address 19
22 is inserted at address 22
5 is already in the table
100 is inserted at address 10
26 is inserted at address 26
8 is inserted at address 8
4 is inserted at address 4
16 is inserted at address 16
5 is in the table
27 is not in the table
45 is not in the table
67 is not in the table
2 is in the table
```