Homework 28

施宇庭 C24066096

Implement a hash table which supports deletion operation

Structure

```
typedef struct Pair {
    int key, item;
} Pair;

typedef struct HashTable {
    Pair** buckets;  // 1D array of Pair*
    int capacity;  // number of buckets
} HashTable;
```

Search

```
Pair* search(HashTable* table, int key) {
    Pair** ht = table->buckets;
    int home = divisionHash(key, table->capacity);
    int curr = home;
    while (ht[curr] && ht[curr]->key != key) {
        curr = (curr + 1) % table->capacity;    // linear probing
        if (curr == home) {
            printf("search %d -> not found\n", key);
            return NULL;
        }
    }
    if (ht[curr] && ht[curr]->key == key) {
        printf("search %d -> %d found\n", key, ht[curr]->item);
        return ht[curr];
    }
    printf("search %d -> not found\n", key);
    return NULL;
}
```

Insert

```
void insert(HashTable* table, Pair* pair) {
    Pair** ht = table->buckets;
    int home = divisionHash(pair->key, table->capacity);
    int curr = home;
    while (ht[curr]) {
```

```
        curr = (curr + 1) % table->capacity;    // linear probing
        if (curr == home) {
            fprintf(stderr, "hash table is full\n");
            return;
        }
    }
    printf("insert %d -> %d inserted\n", pair->key, pair->item);
    ht[curr] = pair;    // if the inserted key is existed before, then
overwrite
}
```

Delete

刪除一個 key-value pair 時，除了將原本的值設為空外，還需要檢查後面是否有共用同一個 hash value 的 key-value pair，如果有就需要將後面的 pairs 往前搬，以保證後面的 pairs 在未來都能夠被搜尋到

```
Pair* delete(HashTable* table, int key) {
    Pair** ht = table->buckets;
    Pair* deleted = NULL;
    int home = divisionHash(key, table->capacity);
    int curr = home;
    while (ht[curr] && ht[curr]->key != key) {
        curr = (curr + 1) % table->capacity;    // linear probing
        if (curr == home) {
            printf("delete %d -> not found\n", key);
            return NULL;
        }
    }
    if (ht[curr] && ht[curr]->key == key) {
        deleted = ht[curr];

        while (ht[curr]) {
            int next = (curr + 1) % table->capacity;    // linear probing
            ht[curr] = ht[next];
            curr = next;
        }
    }
    printf("delete %d -> %d deleted\n", key, deleted->item);
    return deleted;
}
```