1. 程式碼架構

這次作業我採用 LINKED LIST 去寫。Linked list 相較於 Array 的好處我認為主要在 delete 資料時，不需要將 delete 資料後面 index 的 key-value 往前一個一個搬，而是可以將對應的位址刪除，然後將 linked 的連結到下一個節點去，再把對應的 key-value 空間 free 掉，來達成 delete 的步驟。

我通過 structure 將 key value 和 linked list 對應的節點位址寫入，以方便進行操作。再通過 typedef 進行變數重新命名。

```
struct key_value {
    char *key;
    char *value;
    struct key_value *next;
};

typedef struct key_value KV;
typedef KV *link;
```

圖一

此外我還分別寫了五個 function，分別為 deletenode,createlink updateNodeValue,findnode,freelist。來完成這 CRUD。

在 main 裡面有宣告 key value 最多只能輸入 9 個字元，因為有一個存換行。

CRUD 分別為 Create Read Update Delete

因此我認為 RUD 這三個步驟都需要找尋到 linked list node 的位址再去進行相關操作。所以我寫了一個 findnode 的函式。

```
//find data by traverse the node  and return the address of the object.
link findnode(link head, char *key) {
    link ptr = head->next;
    while (ptr != NULL) {
        if (strcmp(ptr->key, key) == 0) {
            return ptr;
        }
        ptr = ptr->next;
    }
    return NULL;
}
```

圖二

Findnode 這個函式是通過將起始位置(只有位址沒有 key-value)，和要尋找的 key 傳入 Function 裡，再從 node1 開始逐一開始找，直到找到對應 key-value 的位址，再傳回位址。

Deletenode 是通過將跳過需要刪除的 node，連接需要刪除的 node 再通過主函示裡的 free，把需要刪除的資料刪除和歸還記憶體。

```
link deletenode(link head, link ptr) {
    link previous = head;
    if (ptr == head) {
        head = head->next;
    } else {
        while (previous->next != ptr) {
            previous = previous->next;
        }
        previous->next = ptr->next;
    }
    return head;
}
```

圖三

Createlink 是將輸入的 key value 存入新創建的 linked list 裡來達成寫入的步驟。

```
link createlink(link head, char *key, char *value) {
    link ptr1 = (link)malloc(sizeof(KV));
    ptr1->next = head->next;
    ptr1->key = strdup(key);
    ptr1->value = strdup(value);
    head->next = ptr1;
    return head;
}
```

圖四

Updatenodevalue 是將新的 value 存入，並過過在主函示呼叫 findnode 找到原本 key 所在的 node 的位置來去修改 value 值

```
void updateNodeValue(link node, char *newValue) {
    free(node->value);
    node->value = strdup(newValue);
    printf("New value: %s\n", newValue);
}
```

圖五

Freelist 是當輸入不為 crud 指令時，把所有動態記憶體歸還

```
void freelist(link head) {
    link ptr1;
    while (head != NULL) {
        ptr1 = head;
        head = head->next;
        //free(ptr1->key);
        //free(ptr1->value);
        free(ptr1);
    }
}
```

圖六

程式輸出結果說明

在主函式一開始以宣告 12 組 key value 值，可以進行 read update delete 等操作。

```
char test_data[TWELVE][2][10] = {
{"KEVIN", "GOOD"}, {"ETHAN", "EXCELLENT"}, {"JIM", "STUDENT"},
{"LEE", "BIGGUY"}, {"KLAY", "SHOOTER"}, {"LAVINE", "DUNKER"},
{"wyf", "handsome"}, {"vivian", "pretty"}, {"thomas", "123453"},
{"CURRY", "SC30"}, {"Kyrie", "211"}, {"james", "king23"}
};
```

圖七

第一步輸入 get KEVIN 可得已輸入的 VALUE。我設定 CRUD 的判斷條件分別為 set、get、upd、del。之後我 set adam 為我的 key，500 為 adam 的 value。
之後通過 get adam 得到正確的 value。
再來用 upd 更新 adam 的 value，通過 get adam 後可以得到新的 value。最後測試 del，再通過 get，可發現 key 找尋不到，因為已被刪除。最後輸入非 crud 指令去結束程式，但需要輸入兩筆數據，因為我目前判斷式是一次輸入兩筆再判斷是否為 crud。以上為我的程式介紹，謝謝。

```
請輸入要得操作
get KEVIN
value equal GOOD
請輸入要得操作
set adam 500
請輸入要得操作
get adam
value equal 500
請輸入要得操作
upd adam 300
New value: 300
請輸入要得操作
get adam
value equal 300
請輸入要得操作
del adam
請輸入要得操作
get adam
Didn't find key
請輸入要得操作
-1 kevin
PS C:\Visual Studio Code\DATASTRUCTURE>
```