HW3

EE3B 110305504 黃聖倫

1. 程式架構

有引用 HW1(SET GET UPDATE DELETE)FUNCTION。分別為

```
void dbUPD(link const head,char *key,char *value);
void dbGET(link const head,char* key);
link dbDEL(link head,char * key);
void dbinsert(link head,char *key,char * value);
```

將原先寫好的 Function 再進行包裝。

針對這次 HW3，六個 FUNCTION(LPUSH LPOP RPUSH RPOP LLEN LRANGE)分別寫出對應 function。

```
void LPUSH(link* head, link* tail, char* key, char* value) {
    link new_node = (link)malloc(sizeof(KV));
    new_node->key = strdup(key);
    new_node->value = strdup(value);
    new_node->prev = NULL;
    new_node->next = *head;

    if (*head != NULL) {
        (*head)->prev = new_node;
    }

    if (*tail == NULL) {
        *tail = new_node;
    }

    *head = new_node;
}
```

圖一

通過引入最新 key value，讓最新 data 變成雙向鏈結的 head。並針對當 head 無資料時，head 和 tail 指向同一點。

```c
void RPUSH(link* head, link* tail, char* key, char* value) {
    link new_node = (link)malloc(sizeof(KV));
    new_node->key = strdup(key);
    new_node->value = strdup(value);
    new_node->prev = *tail;
    new_node->next = NULL;

    if (*tail != NULL) {
        (*tail)->next = new_node;
    }

    if (*head == NULL) {
        *head = new_node;
    }

    *tail = new_node;
}
```

圖二

RPUSH 使用類似寫法，但是將最後輸入的作為 Doubly linked list 的 tail。一樣有針對當原先鏈結無資料時，進行 head=tail=new_node 宣告。

```c
void LPOP(link* head, link* tail, link ptr) {
    link current = NULL;
    if (ptr == NULL) {
        printf("No data can be popped from the left\n");
        return;
    }

    if (*head == ptr) {
        current = *head;
        if (current->next != NULL) {
            *head = current->next;
            (*head)->prev = NULL;
        } else {
            *head = NULL;
            *tail = NULL;
        }
    } else {
        current = ptr;
        if (current->prev != NULL) {
            current->prev->next = current->next;
        }
        if (current->next != NULL) {
            current->next->prev = current->prev;
        }
    }
    printf("%s\n", current->value);
    free(current->key);
    free(current->value);
    free(current);
}
```

圖三

LPOP 為將雙向鏈結最左邊數據 POP 出來，並將 HEAD 位址改為原先 HEAD 的 NEXT。並將輸出資料 FREE 掉，回傳系統動態記憶體。因為我使用將所有數

據傳入相同 doubly linked list，所以原本一定會動到 head，但因為 key 不同，導致可能刪除 doubly linked list 中間位置資料。也就是那個 key 最左邊的位子，如果是這種情況則將 current 的 prev 的 next 指向 current 的 next, current 的 next 的 prev 指向 current 的 prev。

```
void RPOP(link* head, link* tail, link ptr) {
    link current = NULL;
    if (ptr == NULL) {
        printf("No data can be popped from the right\n");
        return;
    }

    if (*tail == ptr) {
        current = *tail;
        if (current->prev != NULL) {
            *tail = current->prev;
            (*tail)->next = NULL;
        } else {
            *head = NULL;
            *tail = NULL;
        }
    } else {
        current = ptr;
        if (current->prev != NULL) {
            current->prev->next = current->next;
        }
        if (current->next != NULL) {
            current->next->prev = current->prev;
        }
    }
    printf("%s\n", current->value);
    free(current->key);
    free(current->value);
    free(current);
}
```

圖四

RPOP 為將雙向鏈結最右邊數據 POP 出來，並將 HEAD 位址改為原先 TAIL 的 prev。並將輸出資料 FREE 掉，回傳系統動態記憶體。因為我使用將所有數據傳入相同 doubly linked list，所以原本一定會動到 tail，但因為 key 不同，導致可能刪除 doubly linked list 中間位置資料。也就是那個 key 最右邊的位子，如果是這種情況則將 current 的 prev 的 next 指向 current 的 next, current 的 next 的 prev 指向 current 的 prev。

```c
void LLEN(link const head,char *key)
{
    link ptr = head;
    int length=0;
    while(ptr!=NULL){
        if(strcmp(ptr->key, key) == 0) length++;
        else length=length;
        ptr=ptr->next;
    }
    printf("Length: %d\n",length);
}
```

圖五

通過在雙向鏈結找相同 key，並增加 length，再輸出為 LLEN。

```c
void LRANGE(link const head, char *key, int start, int stop) {
    if(head==NULL)
    {
        printf("Can't find this key\n");
        return ;
    }
    link ptr = head;
    int index = 0;
    int length = 0;
    // Calculate the length of the list
    while (ptr != NULL) {
        if (strcmp(ptr->key, key) == 0) {
            length++;
        }
        ptr = ptr->next;
    }
    // Handle negative indices
    if (start < 0) {
        start = length + start;
    }
    if (stop < 0) {
        stop = length + stop;
    }
    ptr = head;  // Reset pointer to the beginning
    while (ptr != NULL) {// Print elements within the specified range
        if(length==0)
        {
            printf("Can't find this key");
            break;
        }
        if (strcmp(ptr->key, key) == 0) {
            if (index >= start && index <= stop) {
                printf("%s:%s ", ptr->key, ptr->value);
            }
            index++;
        }
        ptr = ptr->next;
```

LRANGE 會需要輸入三個數據 KEY START END.

因為可能會輸入-1,-2 代表最後一筆數據和倒數第二筆數據。所以我先計算符合 KEY 的數量。之後將 ptr 指向開頭。再用一個判斷式將符合在區間內的 value 印出。以此達成 LRANGE 功能。

程式輸出範例

```
Enter the operation (LPUSH, RPUSH, LPOP, RPOP, LLEN, LRANGE) OR (SET, GET, DEL, UPD): SET CURRY 30
Enter the operation (LPUSH, RPUSH, LPOP, RPOP, LLEN, LRANGE) OR (SET, GET, DEL, UPD): GET CURRY
value equal 30
Enter the operation (LPUSH, RPUSH, LPOP, RPOP, LLEN, LRANGE) OR (SET, GET, DEL, UPD): UPD CURRY 45
New value: 45
Enter the operation (LPUSH, RPUSH, LPOP, RPOP, LLEN, LRANGE) OR (SET, GET, DEL, UPD): GET CURRY
value equal 45
Enter the operation (LPUSH, RPUSH, LPOP, RPOP, LLEN, LRANGE) OR (SET, GET, DEL, UPD): DEL CURRY
Enter the operation (LPUSH, RPUSH, LPOP, RPOP, LLEN, LRANGE) OR (SET, GET, DEL, UPD): GET CURRY
Didn't find key
Enter the operation (LPUSH, RPUSH, LPOP, RPOP, LLEN, LRANGE) OR (SET, GET, DEL, UPD): SET KLAY 11
Enter the operation (LPUSH, RPUSH, LPOP, RPOP, LLEN, LRANGE) OR (SET, GET, DEL, UPD): GET KLAY
value equal 11
Enter the operation (LPUSH, RPUSH, LPOP, RPOP, LLEN, LRANGE) OR (SET, GET, DEL, UPD): GET UNKOWN
Didn't find key
Enter the operation (LPUSH, RPUSH, LPOP, RPOP, LLEN, LRANGE) OR (SET, GET, DEL, UPD): DEL UNKNOWN
Didn't find key
Enter the operation (LPUSH, RPUSH, LPOP, RPOP, LLEN, LRANGE) OR (SET, GET, DEL, UPD): UPD UNKOWN 11
Didn't find key
Enter the operation (LPUSH, RPUSH, LPOP, RPOP, LLEN, LRANGE) OR (SET, GET, DEL, UPD): UPD KLAY 12
New value: 12
Enter the operation (LPUSH, RPUSH, LPOP, RPOP, LLEN, LRANGE) OR (SET, GET, DEL, UPD): GET KLAY
value equal 12
Enter the operation (LPUSH, RPUSH, LPOP, RPOP, LLEN, LRANGE) OR (SET, GET, DEL, UPD): DEL KLAY
Enter the operation (LPUSH, RPUSH, LPOP, RPOP, LLEN, LRANGE) OR (SET, GET, DEL, UPD): GET KLAY
Didn't find key
```

此為測試 HW1。輸入 CURRY 30，再分別進行 GET UPD DEL，可看出得到預計結果。之後 SET KLAY 11。經過 GET 發現也無問題。之後嘗試 GET 一個不存在的 KEY，發現無法找到 KEY，因此無法作對應操作。之後再嘗試 UPD KLAY，再 GET，得到新的 VALUE。之後 DEL KLAY 再 GET，發現找不到。
以上為測試 HW1 之測資。

LIST 測資

```
Enter the operation (LPUSH, RPUSH, LPOP, RPOP, LLEN, LRANGE) OR (SET, GET, DEL, UPD): LPUSH AB 123 234 3245 2345
Enter the operation (LPUSH, RPUSH, LPOP, RPOP, LLEN, LRANGE) OR (SET, GET, DEL, UPD): LPOP AB
2345
Enter the operation (LPUSH, RPUSH, LPOP, RPOP, LLEN, LRANGE) OR (SET, GET, DEL, UPD): RPOP AB
123
Enter the operation (LPUSH, RPUSH, LPOP, RPOP, LLEN, LRANGE) OR (SET, GET, DEL, UPD): LLEN AB
Length: 2
Enter the operation (LPUSH, RPUSH, LPOP, RPOP, LLEN, LRANGE) OR (SET, GET, DEL, UPD): LRANGE AB 0 1
AB:3245 AB:234
```

先使用 LPUSH 輸入 KEY 為 AB，VALUE 為 123 234 3245 2345，四個數據。
使用 LPOP，得最後輸入資料 2345。再使用 RPOP 得最早輸入資料 123。
之後使用 LLEN，原先輸入四筆資料，提出兩筆，所以只剩兩筆。
再使用 LRANGE，輸入起始和終點得 3245 234。與預計相同。

```
Enter the operation (LPUSH, RPUSH, LPOP, RPOP, LLEN, LRANGE) OR (SET, GET, DEL, UPD): RPUSH AB 4354 1231
Enter the operation (LPUSH, RPUSH, LPOP, RPOP, LLEN, LRANGE) OR (SET, GET, DEL, UPD): LPOP AB
3245
Enter the operation (LPUSH, RPUSH, LPOP, RPOP, LLEN, LRANGE) OR (SET, GET, DEL, UPD): RPOP AB
1231
Enter the operation (LPUSH, RPUSH, LPOP, RPOP, LLEN, LRANGE) OR (SET, GET, DEL, UPD): LLEN AB
Length: 2
Enter the operation (LPUSH, RPUSH, LPOP, RPOP, LLEN, LRANGE) OR (SET, GET, DEL, UPD): LRANGE AB
0 -1
AB:234 AB:4354
```

之後使用 RPUSH 輸入 4354 1231。再使用 LPOP，得原先 LPUSH 最後輸入資料 3245(因為 2345 已提出)。再使用 RPOP 得 RPUSH 最後輸入資料 1231。
之後測試 LLEN 得到 2(原先兩筆，輸入兩筆提出兩筆，所以為 2)。之後測試 LRANGE 輸出 LPUSH 唯一沒被提出資料 234，和 RPUSH 最先輸入資料 4354。
可知結果與預期相同。

```
Enter the operation (LPUSH, RPUSH, LPOP, RPOP, LLEN, LRANGE) OR (SET, GET, DEL, UPD): LPOP Ac
No data can be popped from the left
Enter the operation (LPUSH, RPUSH, LPOP, RPOP, LLEN, LRANGE) OR (SET, GET, DEL, UPD): RPOP AC
No data can be popped from the right
Enter the operation (LPUSH, RPUSH, LPOP, RPOP, LLEN, LRANGE) OR (SET, GET, DEL, UPD): LPUSH AC 213 435 123
Enter the operation (LPUSH, RPUSH, LPOP, RPOP, LLEN, LRANGE) OR (SET, GET, DEL, UPD): LPOP AC
123
Enter the operation (LPUSH, RPUSH, LPOP, RPOP, LLEN, LRANGE) OR (SET, GET, DEL, UPD): LPOP AB
234
Enter the operation (LPUSH, RPUSH, LPOP, RPOP, LLEN, LRANGE) OR (SET, GET, DEL, UPD): RPUSH CDE 324 45 1234
Enter the operation (LPUSH, RPUSH, LPOP, RPOP, LLEN, LRANGE) OR (SET, GET, DEL, UPD): LPOP CDE
324
Enter the operation (LPUSH, RPUSH, LPOP, RPOP, LLEN, LRANGE) OR (SET, GET, DEL, UPD): LLEN CDE
Length: 2
Enter the operation (LPUSH, RPUSH, LPOP, RPOP, LLEN, LRANGE) OR (SET, GET, DEL, UPD): LLEN AC
Length: 2
Enter the operation (LPUSH, RPUSH, LPOP, RPOP, LLEN, LRANGE) OR (SET, GET, DEL, UPD): LPOP AB
4354
Enter the operation (LPUSH, RPUSH, LPOP, RPOP, LLEN, LRANGE) OR (SET, GET, DEL, UPD): LPOP AB
No data can be popped from the left
```

之後輸入不同 KEY，來測試。
使用 LPOP 加上不存在 KEY，發現沒有資料。使用 RPOP 也一樣沒有資料。
之後使用 LPUSH，輸入 KEY=AC， VALUE= 213 435 123。
通過 LPOP AC，得最後 PUSH 的 DATA 123。再使用 LPOP AB，得第一次 LPUSH 剩餘資料 234。
之後再嘗試使用 RPUSH 第三個 KEY=CDE，VALUE 分別為 324 45 1234。LPOP CDE, LLEN CDE，可發現與預期相同，得最後輸入資料，加上輸入資料數-1 的長度。
之後測試 LLEN AC，因為原先輸入三筆，提出一筆後還有兩筆(LLEN=2)
再使用 LPOP 把 AB 剩餘資料提出(變成無資料)
再使用一次發現得不到 VALUE。

以上為我的測資，可發現結果都與預期相同。