國立成功大學 112 學年度第一學期
程式設計(一) - 期末考 (上機測驗)

2023 Fall NCKU Program Design I Final

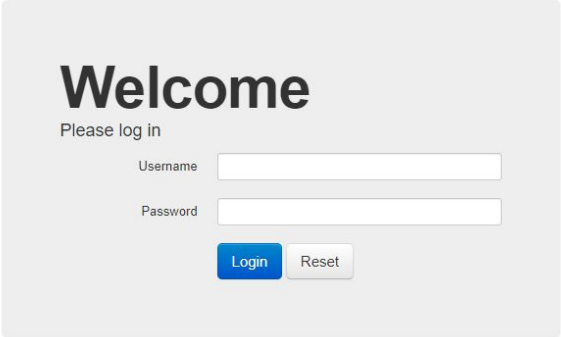
Before Start

- 不要攻擊我們的系統，否則我們會直接把你當掉
- 考試結束後，我們會將所有同學的程式碼抓進比對系統，如有發現疑似作弊的情形，除了本次考試 0 分之外，比照校規處理，請同學注意
- Judge 為及時回饋制，你可以在每一次上傳後得知你在該題組獲得了幾分

Before Start

- Do not attack our system, otherwise, you will fail in this course.
- After the exam, we will take all students' codes into a comparison system. If we find any suspected cheating, in addition to getting zero in this exam, it will be dealt with according to the school rules. Please pay attention to this.
- The Judge is a real-time feedback system, and you can know how many points you scored in that set of problems after each upload.

System



- Contest Management System (CMS)
- It is an Online Judge System
- IP: 140.116.154.71
  - Please visited this address by any browser (do not use ssh)
- Account: Your Student ID
- Password: Your Password

Final Problem

- 在作業 5 中，你開了一間卡比商店，這家卡比商店因為你寫的程式導致很多觀光客慕名而來，店裡到店外滿滿都是排隊的隊伍，為了有效的管理隊伍的秩序，以及即使的對隊伍做一些控制，你必須要設計一個程式來模擬隊伍當前的狀況
- In Assignment 5, you opened a Kirby store. This Kirby store has attracted many tourists due to the program you wrote. The store is filled with lines of customers both inside and outside. To effectively manage the order of the queue and to control it in a timely manner, you must design a program to simulate the current situation of the queue.

Final Problem

- 已知我們這個程式必須將隊伍用一個 LinkedList 表示，這一個 LinkedList 的結構如下：

```
 7 struct LinkedList{
 8     char name[20];
 9     int passwd_public[10][10][10][10];
10     int*** passwd_private;
11     struct LinkedList* back_head;
12     struct LinkedList* next;
13 };
```

Final Problem

- We know that our program must represent the queue using a LinkedList. The structure of this LinkedList is as follows:

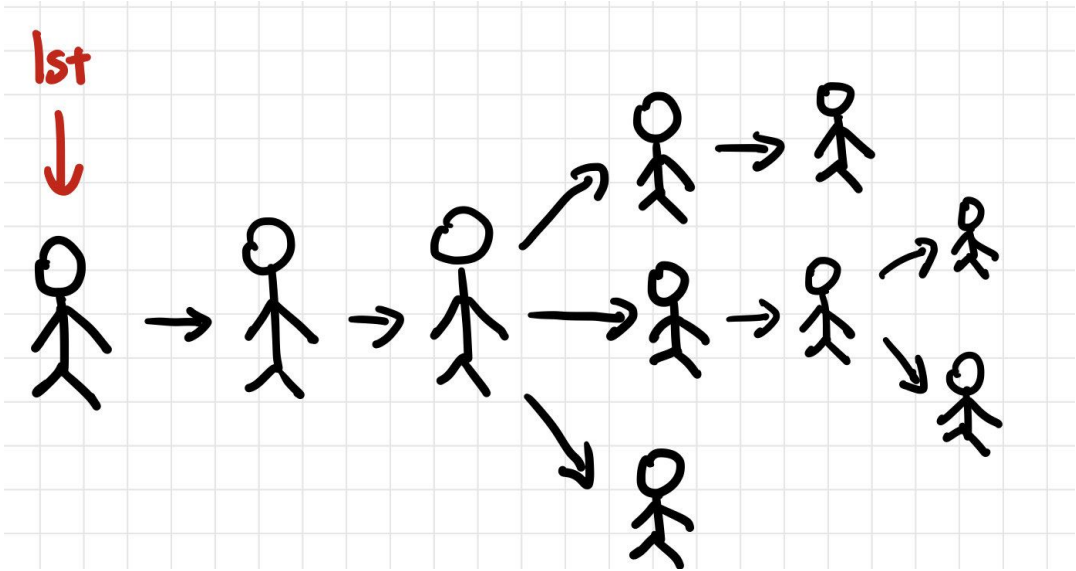
```
 7 struct LinkedList{
 8     char name[20];
 9     int passwd_public[10][10][10][10];
10     int*** passwd_private;
11     struct LinkedList* back_head;
12     struct LinkedList* next;
13 };
```

LinkedList Graph

- 由於現場人數眾多，隊伍不一定是一條直線，有可能同時有很多人在某一個人的後面
- Due to the large number of people on site, the queue may not necessarily be a straight line. It's possible that there are many people behind a single person at the same time.

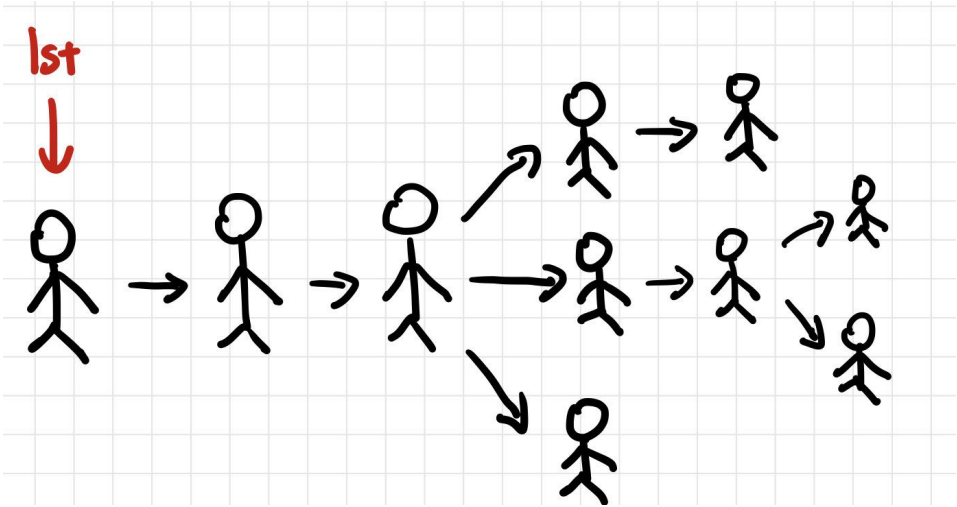
LinkedList Graph

- 特別注意，箭頭的意思表示被指向的人在後面
- Pay special attention, the arrow indicates that the person being pointed to is behind.



LinkedList Graph

- 以此圖為例，第三個人的後面就同時排了 3 個人
- As an example in this diagram, there are 3 people lined up simultaneously behind the third person.



LinkedList member: name

- name 表示該位置的人的名字
- name (variable) represents the name of the person at that position.

```
 7 struct LinkedList{
 8     char name[20];
 9     int passwd_public[10][10][10][10];
10     int*** passwd_private;
11     struct LinkedList* back_head;
12     struct LinkedList* next;
13 };
```

LinkedList member: passwd_public

- passwd_public 表示該人的會員帳號
- passwd_public represents the person's membership account.

```
 7 struct LinkedList{
 8     char name[20];
 9     int passwd_public[10][10][10][10];
10     int*** passwd_private;
11     struct LinkedList* back_head;
12     struct LinkedList* next;
13 };
```

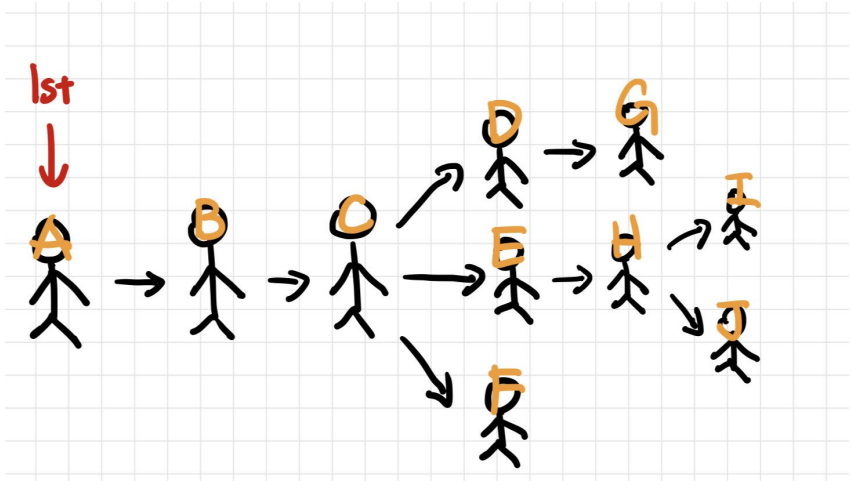
LinkedList member: passwd_private

- passwd_private 表示該人的會員密碼
- passwd_private represents the person's membership password.

```
 7 struct LinkedList{
 8     char name[20];
 9     int passwd_public[10][10][10][10];
10     int*** passwd_private;
11     struct LinkedList* back_head;
12     struct LinkedList* next;
13 };
```

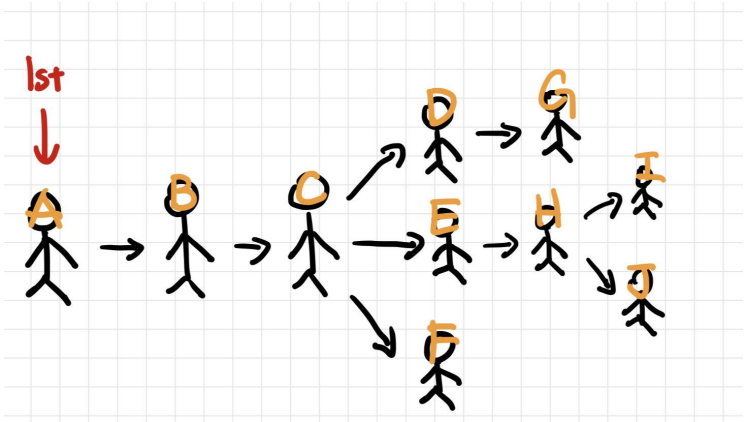
LinkedList member: back_head

- back_head 表示排在該人後面最左邊的人的指標
- back_head represents the pointer to the person who is at the far left behind the given person.
- A -> backhead = B
- C -> backhead = F
- D -> backhead = G
- H -> backhead = J
- F -> backhead = NULL



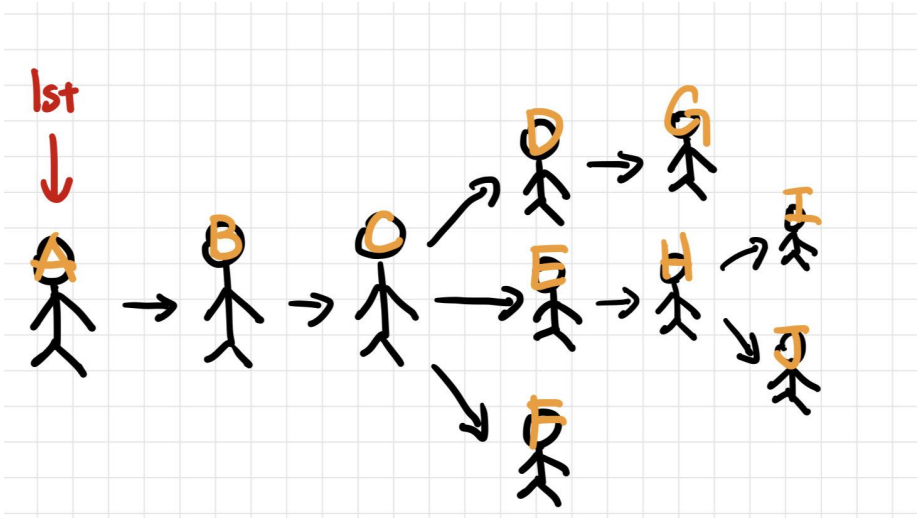
LinkedList member: next

- next 表示排在該人右邊的第一個人
- next represents the first person to the right of the given person.
- 特別注意，以下圖為例：
- 由於 G, H 排在不同人的後面
  因此我們不會認定 G 在 H 的右邊
- Since G and H are lined up
  behind different people, we do
  not consider G to be on the right of H.



LinkedList member: next

- A -> next = NULL
- C -> next = NULL
- F -> next = E
- E -> next = D
- D -> next = NULL
- H -> next = NULL
- J -> next = I



System Testing

- 在實際測試的時候，我們的 Loader Code 會添加 validate function 將你的 LinkedList 結構全部檢查一遍，確保所有的資訊都是正確的
- Loader Code 可上 moodle 下載
- During actual testing, our Loader Code will add a validate function to thoroughly check your LinkedList structure, ensuring that all information is correct.
- The Loader Code can be downloaded from the CMS system or Moodle.

Submit Code

- 舉例來說，下方是 create function 的模板
- 你只需要把 create function 實作完即可上傳到 create 的題組

```c
#include <stdlib.h>
#include <string.h>
#include <stdio.h>

struct LinkedList;

struct LinkedList{
    char name[20];
    int passwd_public[10][10][10][10];
    int*** passwd_private;
    struct LinkedList* back_head;
    struct LinkedList* next;
};

struct LinkedList* create(char target[20],int p1,int n,int m,int l,int p2)
{
    // Write your code here!
}
```

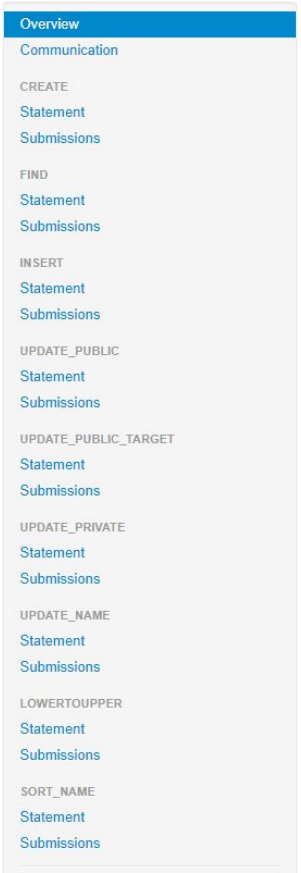
Submit Code

- For example, below is the template for the create function.
- You only need to implement the create function and then upload it to the problem set for create.

```c
#include <stdlib.h>
#include <string.h>
#include <stdio.h>

struct LinkedList;

struct LinkedList{
    char name[20];
    int passwd_public[10][10][10][10];
    int*** passwd_private;
    struct LinkedList* back_head;
    struct LinkedList* next;
};

struct LinkedList* create(char target[20],int p1,int n,int m,int l,int p2)
{
    // Write your code here!
}
```

Submit

- Submissions -> Choose File -> Submit



Submit

- click detail to check your result

Time	Status		Score
3:27:02 AM	Evaluated	details	10 / 10

Loader Code

- 你必須要實作許多 Function 來對這些隊伍做一些操作：
  - create
  - find
  - insert
  - update_public
  - update_public_target
  - update_name
  - LowerToUpper
  - sort_name

Loader Code

- You must implement many functions to perform various operations on these queues:
  - create
  - find
  - insert
  - update_public
  - update_public_target
  - update_name
  - LowerToUpper
  - sort_name

Loader Code

- 每一個題組的最後都有一個模板可以參考，上傳時只需要傳該 Function 的實作與其他 struct 與 header file 即可，請不要把所有 Function 都實作好一次傳上來，否則會 Compiler Error
- At the end of each set of problems, there is a template for reference. When uploading, you only need to submit the implementation of that specific function along with other structs and header files. Please do not implement all functions at once and upload them together, as this will cause a Compiler Error.

struct LinkedList* create (5 %)

- 參數：(char target[20],int p1,int n,int m,int l,int p2)
- 創建一個新的點
- 將名字設定成 target
- 將會員編號 (passwd_public) 的所有 index 的值都設定成 p1
- 將會員密碼 (passwd_private) 動態分配記憶體成一個 n * m * l 的三維陣列，並將所有位置的值設定成 p2
- 回傳設定好的 pointer

struct LinkedList* create (5 %)

- Parameters: (char target[20], int p1, int n, int m, int l, int p2)
- Create a new node.
- Set the name to target.
- Set all indices of the membership number (passwd_public) to the value p1.
- Dynamically allocate memory for the membership password (passwd_private) as a n * m * l three-dimensional array, and set the value of all positions to p2.
- Return the configured variable.

struct LinkedList* create (5 %)

```c
#include <stdlib.h>
#include <string.h>
#include <stdio.h>

struct LinkedList;

struct LinkedList{
    char name[20];
    int passwd_public[10][10][10][10];
    int*** passwd_private;
    struct LinkedList* back_head;
    struct LinkedList* next;
};

struct LinkedList* create(char target[20],int p1,int n,int m,int l,int p2)
{
    // Write your code here!
}
```

struct LinkedList* find (10 %)

- 參數：(struct LinkedList* head,char target[20])
- 找到 target 所在的記憶體位置並回傳
- 如果找不到則回傳 NULL
- Parameters: (struct LinkedList* head, char target[20])
- Find and return the memory location of target.
- If it cannot be found, return NULL.

struct LinkedList* find (10 %) - Hints

- 妥善利用 back_head 與 next 並搭配遞迴的方式找到 target
- Make proper use of back_head and next, and employ recursion to find target.

struct LinkedList* find (10 %)

```c
#include <stdlib.h>
#include <string.h>
#include <stdio.h>

struct LinkedList;

struct LinkedList{
    char name[20];
    int passwd_public[10][10][10][10];
    int*** passwd_private;
    struct LinkedList* back_head;
    struct LinkedList* next;
};

struct LinkedList* find(struct LinkedList* head,char target[20])
{
    // Write your code here!
}
```

int insert (10 %)

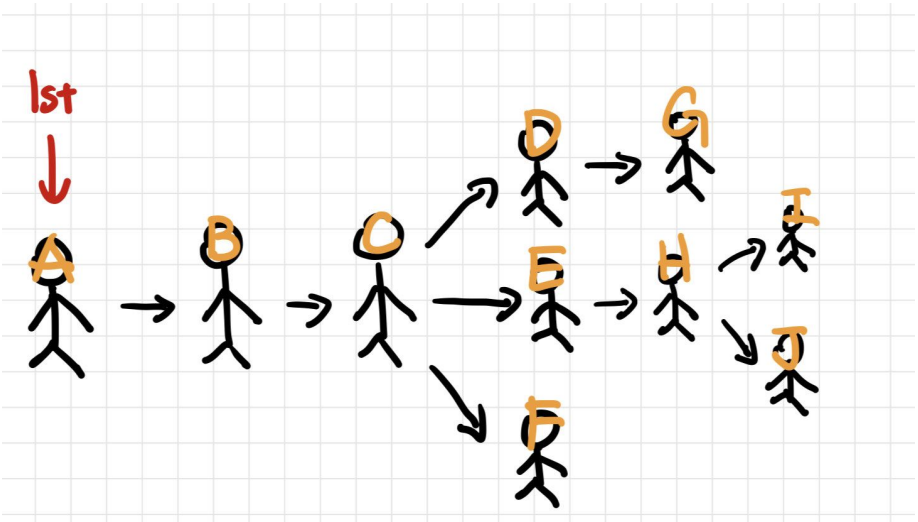
- 參數：(struct LinkedList* head,char target[20],struct LinkedList* insert_people)
- 把 insert people 加入到 target 的後面
  - 如果 target 的後面已經有人，則加入到 target 後面的所有人當中，最右邊的位置
- 如果隊伍中沒有 target 這個人，請回傳 -1
- 否則，請插入完後回傳 0

int insert (10 %)

- Parameters: (struct LinkedList* head, char target[20], struct LinkedList* insert_people)
- Add insert_people behind the target.
- If there are already people behind the target, then add to the far right position among all the people behind the target.
- If there is no person named target in the queue, please return -1.
- Otherwise, after completing the insertion, return 0.

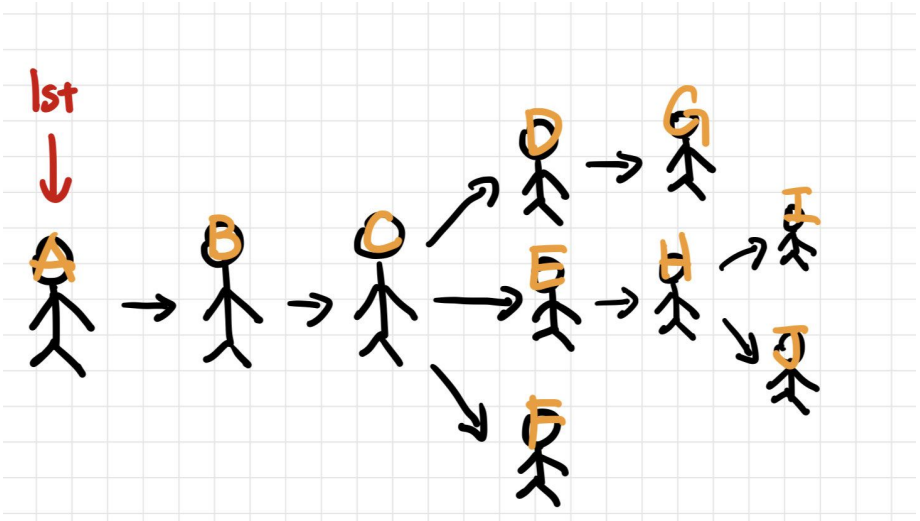
int insert (10 %)

- 假設目前有一個人 K 要加入到 C 的後面
- 但因為 C 目前後面有人了 所以他會被放到最右邊去
- 也就是 D 的右邊



int insert (10 %)

- Assume that currently a person K wants to join behind C.
- However, since there are already people behind C, K will be placed at the far right.
- That is, to the right of D.



int insert (10 %) - Hints

- 參數會給你 head 就表示你必須想辦法依靠 head 找到 target 在哪
- 所以你會需要 find function 輔助你完成 insert
- 特別注意到兩種 Case：
  - Case 1: 後面沒人 -> back_head 更新
  - Case 2: 後面有人 -> 用 back_head 找到最右邊的人更新 next

int insert (10 %) - Hints

- The parameter head implies that you must find a way to locate target using head. Therefore, you will need a find function to assist you in completing the insert.
- Pay special attention to two cases:
  - Case 1: No one is behind -> Update back_head.
  - Case 2: There are people behind -> Use back_head to find the person on the far right and update next.

int insert (10 %)

```c
#include <stdlib.h>
#include <string.h>
#include <stdio.h>

struct LinkedList;

struct LinkedList{
    char name[20];
    int passwd_public[10][10][10][10];
    int*** passwd_private;
    struct LinkedList* back_head;
    struct LinkedList* next;
};

int insert(struct LinkedList* head,char target[20],struct LinkedList* insert_people)
{
    // Write your code here
}
```

int update_public (5 %)

- 參數：(struct LinkedList* head,char target[20],int idx1,int idx2,int idx3,int idx4,int* val)
- 將 target 的會員編號的索引 [idx1][idx2][idx3][idx4] 位置更改成 val
- 並且把 val 加上原本 [idx1][idx2][idx3][idx4]的值
- 如果發生 index 出界的情形（ >= 10 or < 0 ）不須做任何動作回傳 -2
- 如果找不到 target 則不需要做任何動作回傳 -1
- 更改成功則回傳 0

int update_public (5 %)

- Parameters: (struct LinkedList* head, char target[20], int idx1, int idx2, int idx3, int idx4, int* val)
- Change the value at the index [idx1][idx2][idx3][idx4] of the member number of 'target' to 'val', and add 'val' to the original value at [idx1][idx2][idx3][idx4].
- If an index out of bounds occurs (>= 10 or < 0), do nothing and return -2.
- If 'target' is not found, do nothing and return -1.
- If the change is successful, return 0.

int update_public (5 %)

- 舉例來說：如果呼叫 update_public(head,"Colten",2,3,1,4,9)
- 且原本 Colten 的會員編號[2][3][1][4] = 20
- 那你必須把[2][3][1][4] 更改成 9
- 並把 val 更改成 20 + 9 = 29
- For instance, if the function update_public(head, "Colten", 2, 3, 1, 4, 9) is called, and originally the member number of 'Colten' at index [2][3][1][4] equals 20, then you must change the value at [2][3][1][4] to 9, and update 'val' to be 20 + 9 = 29.

int update_public (5 %)

```c
#include <stdlib.h>
#include <string.h>
#include <stdio.h>

struct LinkedList;

struct LinkedList{
    char name[20];
    int passwd_public[10][10][10][10];
    int*** passwd_private;
    struct LinkedList* back_head;
    struct LinkedList* next;
};

int update_public(struct LinkedList* head,char target[20],int idx1,int idx2,int idx3,int idx4,int* val)
{
    // Write your code here
}
```

int update_private (3 %)

- 參數：(struct LinkedList* head,char target[20],int idx1,int idx2,int idx3,int val)
- 將 target 的會員密碼索引[idx1][idx2][idx3] 更改成 val
- 如果找不到 target 則回傳 -1
- 否則回傳 0

int update_private (3 %)

- Parameters: (struct LinkedList* head, char target[20], int idx1, int idx2, int idx3, int val)
- Change the value of the member password at index [idx1][idx2][idx3] for 'target' to 'val'.
- If 'target' is not found, return -1.
- Otherwise, return 0.

int update_private (3 %)

```c
#include <stdlib.h>
#include <string.h>
#include <stdio.h>

struct LinkedList;

struct LinkedList{
    char name[20];
    int passwd_public[10][10][10][10];
    int*** passwd_private;
    struct LinkedList* back_head;
    struct LinkedList* next;
};

int update_private(struct LinkedList* head,char target[20],int idx1,int idx2,int idx3,int val)
{
    // Write your code here!
}
```

void update_public_target (5 %)

- 參數：本 Function 的參數請自行填入 (參考 Loader Code)
- 此 Function 將會有 6 個參數
- 將第一個參數的索引[參數2][參數3][參數4][參數5] 更改成 參數6
- 保證索引不會出界

void update_public_target (5 %)

- Parameters: Please fill in the parameters for this function yourself (refer to the Loader Code).
- This function will have 6 parameters.
- Change the index [Parameter 2][Parameter 3][Parameter 4][Parameter 5] of the first parameter to Parameter 6.
- It is guaranteed that the index will not be out of bounds.

void update_public_target (5 %)

```c
#include <stdlib.h>
#include <string.h>
#include <stdio.h>

struct LinkedList;

struct LinkedList{
    char name[20];
    int passwd_public[10][10][10][10];
    int*** passwd_private;
    struct LinkedList* back_head;
    struct LinkedList* next;
};

void update_public_target(// 自行填入)
{
    // Write your code here
}
```

int update_name (2 %)

- 參數：(struct LinkedList* head,char target[20],char modify[20])
- 先找到 target 的所在位置
- 並將該位置上的名字更改成 modify
- 更改成功回傳 0
- 找不到 target 這一個人則回傳 -1

int update_name (2 %)

- Parameters: (struct LinkedList* head, char target[20], char modify[20])
- First, find the location of 'target'.
- Then, change the name at that location to 'modify'.
- If the change is successful, return 0.
- If 'target' is not found, return -1.

int update_name (2 %)

```c
#include <stdlib.h>
#include <string.h>
#include <stdio.h>

struct LinkedList;

struct LinkedList{
    char name[20];
    int passwd_public[10][10][10][10];
    int*** passwd_private;
    struct LinkedList* back_head;
    struct LinkedList* next;
};

int update_name(struct LinkedList* head,char target[20],char modify[20])
{
    // Write your code here!
}
```

int LowerToUpper (5 %)

- 參數：(struct LinkedList* head,char target[20])
- 先找到 target 的所在位置，並將他改名
- 改名的方式為，將此人的名字小寫全部換成大寫
- Ex. Colten -> COLTEN
- 更改成功回傳 0
- 找不到 target 則回傳 -1

int LowerToUpper (5 %)

- Parameters: (struct LinkedList* head, char target[20])
- First, locate the position of 'target', and then change their name. The method of changing the name is to convert all lowercase letters in the person's name to uppercase.
- For example, Colten -> COLTEN.
- If the change is successful, return 0.
- If 'target' is not found, return -1.

int LowerToUpper (5 %)

```c
#include <stdlib.h>
#include <string.h>
#include <stdio.h>

struct LinkedList;

struct LinkedList{
    char name[20];
    int passwd_public[10][10][10][10];
    int*** passwd_private;
    struct LinkedList* back_head;
    struct LinkedList* next;
};

int LowerToUpper(struct LinkedList* head,char target[20])
{
    // Write your code here!
}
```

void sort_name (5 %)

- 參數：(char target[40])
- 將 target 照字典序 (ASCII Code 的大小由小到大) 排序好即可
- Hints: 由於傳入 target 的方式是 pass by reference，所以你在這個 Function 把 target 排序好是會連帶影響到 Loader Code 的
- 如果你不會使用 qsort，你可以利用氣泡排序慢慢交換

```
sort Colten
After Sorting: Celnot
sort ABCD
After Sorting: ABCD
```

void sort_name (5 %)

```
sort Colten
After Sorting: Celnot
sort ABCD
After Sorting: ABCD
```

- Parameters: (char target[40])
- Sort 'target' in dictionary order (ascending order of ASCII codes).
- Hints: Since 'target' is passed by reference, sorting it within this function will also affect the Loader Code. If you're not familiar with using qsort, you can slowly swap using bubble sort.

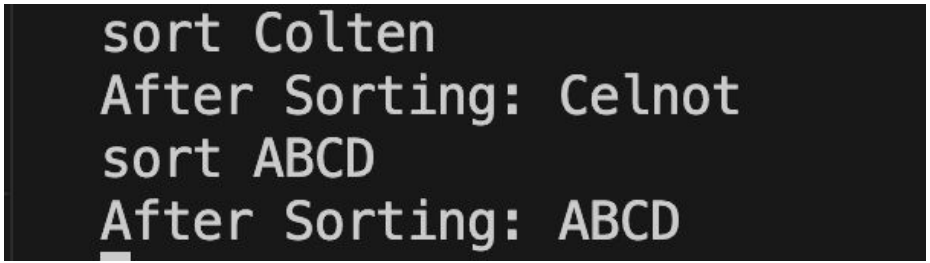
void sort_name (5 %)

- 特別注意：不用實際找到 target 這個節點更改名字
- 就單純把 target 排序好就可以了

```
sort Colten
After Sorting: Celnot
sort ABCD
After Sorting: ABCD
```

void sort_name (5 %)

- There is no need to actually find the node named 'target' to change the name. Simply sort 'target' properly and that will suffice.



void sort_name (5 %)

```c
#include <stdlib.h>
#include <string.h>
#include <stdio.h>

struct LinkedList;

struct LinkedList{
    char name[20];
    int passwd_public[10][10][10][10];
    int*** passwd_private;
    struct LinkedList* back_head;
    struct LinkedList* next;
};


// 你使用 qsort 應該會還需要寫一個 compare function，可以直接寫在這

void sort_name(char target[40])
{
    // Write your code here!
}
```

Scoring

- Total: 50 %
  - create: 5 %
  - find: 10 %
  - insert: 10 %
  - update_public: 5 %
  - update_public_target: 5 %
  - update_private: 3 %
  - update_name: 2 %
  - LowerToUpper : 5 %
  - sort_name: 5 %

Function Help

- 所有題組裡面，除了 find 題組之外，我們都有直接提供 find 函數給大家使用，避免你們寫不出 find 就拿不到分數
  - 但缺點是，這樣你們沒辦法在本機端自己測試，只能直接傳上來 judge
- 使用 Function Help 分數不會打折，也不會扣分，請善加利用
- 但如果你想要自己寫 find function 請不要將其命名成 find，避免 Compiler Error，請命名成 find2 之類的

Function Help

- In all problem sets, except for the 'find' problem set, we have directly provided a 'find' function for everyone to use, to avoid situations where you cannot receive points because you are unable to write the 'find' function yourself.
- However, the downside is that you will not be able to test it on your own local machine and can only upload it to the judge system.

Function Help

- Using the 'Function Help' will not result in a deduction of points or a discount on your score, so please make good use of it. However, if you wish to write your own 'find' function, please do not name it 'find' to avoid a Compiler Error. Instead, name it something like 'find2'.

Function Help

- find 函數的使用方法跟 find 題組的參數定義與回傳值一模一樣
- 可以直接在非 find 的題組使用
- The usage of the find function is identical to the parameter definitions and return values of the find problem set. It can be used directly in problem sets other than the find set.

Testing Data

- Download from moodle
- Limit:
  - 保證 LinkedList 節點數量不會超過 20 個
  - 保證 private_passwd 的三個維度不超過 20
  - 保證所有跟 passwd 有關的值不超過 10000
  - name 只有英文字母大小寫，且長度不超過 19
  - 操作次數不超過 50
  - sort 給的字串長度不超過 39
  - 保證 update private 的時候不會發生索引出界

Testing Data

- It is guaranteed that the number of nodes in the LinkedList will not exceed 20.
- It is guaranteed that the three dimensions of private_passwd will not exceed 20.
- It is guaranteed that all values related to passwd will not exceed 10,000.
- The name consists only of upper and lower case English letters, and the length does not exceed 19 characters.
- The number of operations will not exceed 50.
- The length of the string given to sort will not exceed 39 characters.
- It is guaranteed that there will be no index out of bounds during the update of private.

Testing Data

- 第一筆測試資料都是 sample.in
- 假設你目前寫的是 create function，那我們測試時會將助教的正解單獨拿掉 create function，改成使用你的 create function 來測試你寫的對不對
- The first set of test data is always "sample.in".
- Suppose you are currently writing the create function; during our testing, we will remove the teaching assistant's correct implementation of the create function and replace it with your create function to test if what you have written is correct.

Hints

- 你可以想像成，每一個節點後面都是一個 LinkedList
- You can think of it as each node being followed by a LinkedList.
- LinkedList of LinkedList