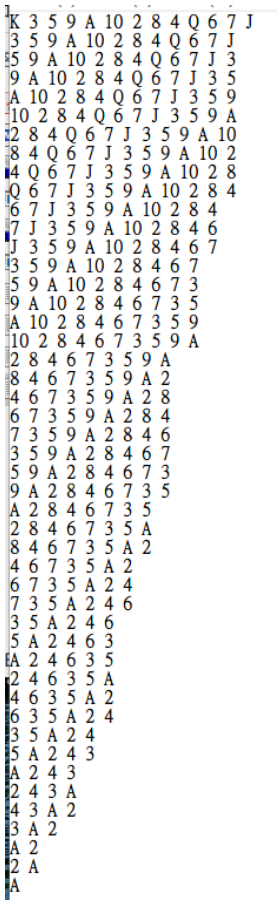
一、執行結果

Part1

```
PS D:\Users\User\Desktop\DS_hw3> Get-Content .\p1_input.txt | .\hw3_1.exe > .\p1_output.txt
```



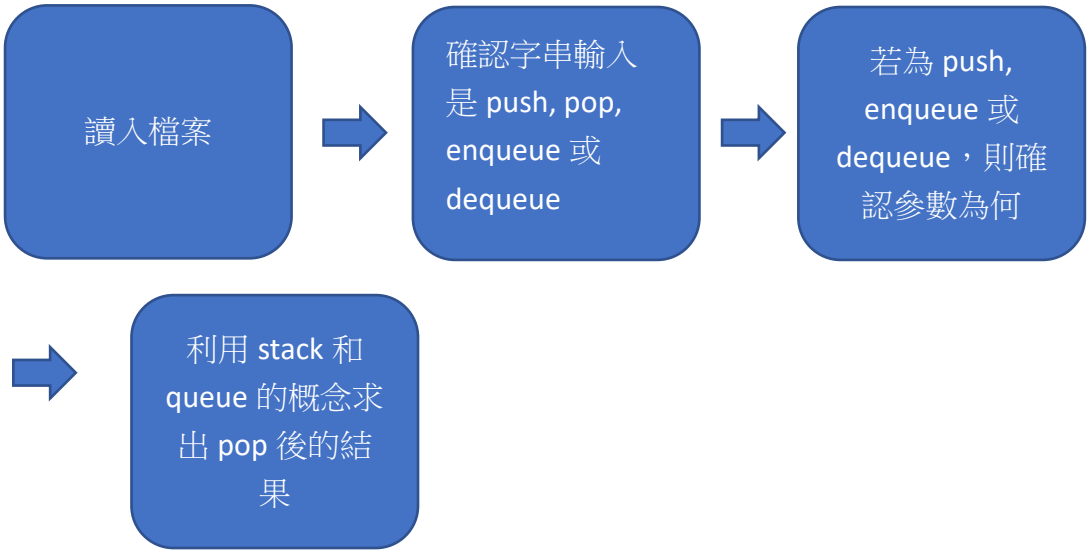
Part2

```
PS D:\Users\User\Desktop\DS_hw3> Get-Content .\p2_input.txt | .\hw3_2.exe > .\p2_output.txt
```

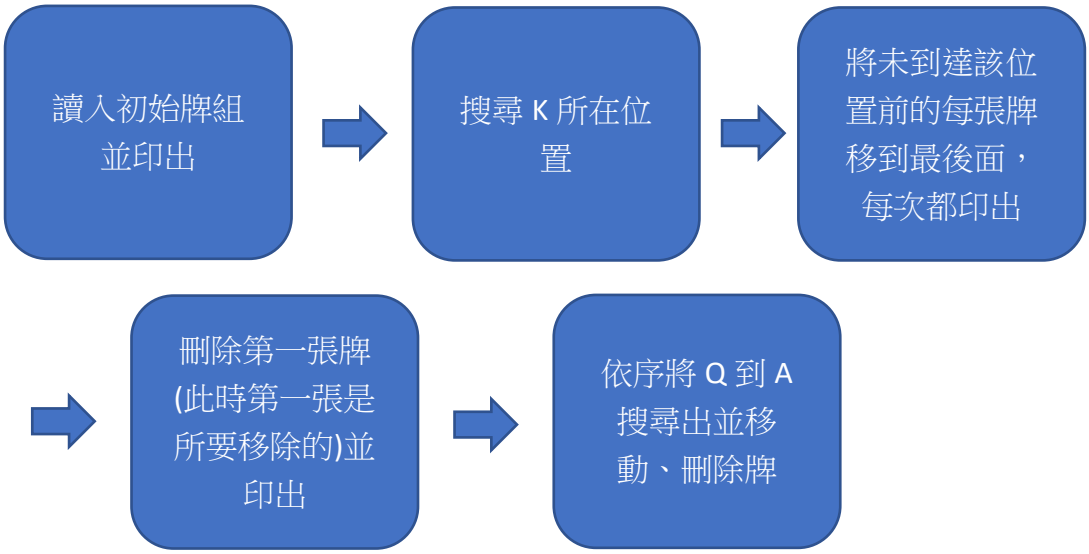
二、流程圖

Part1

與上週作業一樣，但將 stack 和 queue 用 linked list 表示後，不用設定容量，要新增和刪除資料時也相當方便



Part2



三、函式說明

Part1

void push(int data)

- top 原本在-1，代表空 stack。Push 一個資料進來後，top 就往上疊，跑到 0，同時在那個位置儲存資料。當 top 到達最大容量-1 時(因為以 0 作為起點，故最後一個位置是最大容量-1)，就印出 stack 已滿的訊息。

int pop()

- 先用一個變數儲存目前位於 top 的資料(也就是待會要被 pop 的資料)，接著讓 top 減 1，無視掉原 top 的資料，視同將它 pop 出去。最後將儲存原 top 資料的變數回傳，即得想要結果。

void enqueue(char data)

- 一開始 front 和 rear 都是 -1，當 enqueue 一個資料進來，rear 就加 1 跑到 0，同時在該位置儲存資料。當 rear 達到最大容量-1 時，印出 queue 已滿。

char dequeue()

- 先將 front 加 1，用一個變數儲存新 front 位置的資料(即為等下要被 dequeue 的資料)，再將變數回傳。

Part2

void printList(FILE* fout, int i)

- 若 first==NULL，代表 list 為空。其餘情況，將 tmp 設為 first，在 tmp==NULL 也就是 list 到達最後之前，將每個 tmp->point 印出。印出後 tmp 指到原本那個節點的 next。

pokerPointer searchNode(char toFind[2])

- toFind 是傳入的要搜尋的字串，本題是要找的撲克牌點數。在 tmp==NULL 也就是 list 到達最後之前，比對每個 tmp->point 和 toFind 的相同性。當比到一樣時，將此時 tmp 所在位置回傳。若不同，tmp 指到原本那個節點的 next。

pokerPointer deleteNode(pokerPointer pos)

- 若 first==NULL，代表 list 為空。其餘情況，將 first 指到 first 的 next，代表新的 first，再將 pos 的記憶體空間釋出。因為 pos 紀錄原本 first 所指位置。最後回傳新 first 的位置。

四、設計討論

這次 part1 將上次的不足大致改正了，part2 的部分基本上就是 linked list 的實作。由於前幾次作業在 getopt 和 get-content 上有很多問題，再加上 c 語言本身的讀寫檔,fscanf, fgets 以及字串比對、複製等等都還蠻不熟的。這次就花了很多時間在這上面，希望這次有真的搞懂了。