Data Structure Assignment 3

ID: E14066282	**Name:** 溫梓傑	**Department:** ME 110

○ Result Screenshots

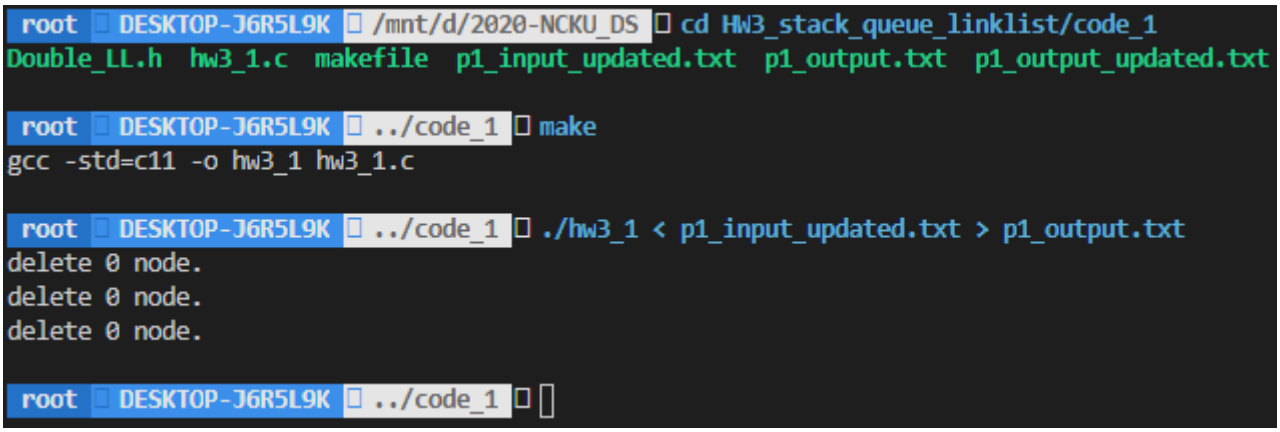

Figure 1 Screenshot of command line (hw3_1)

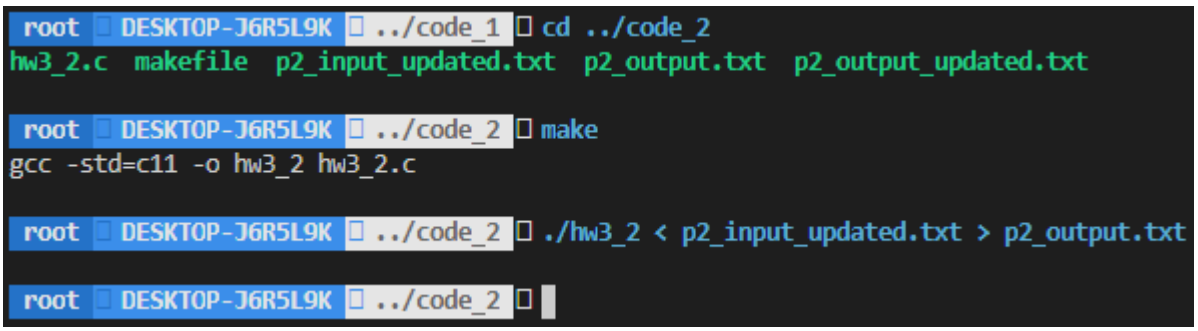

Figure 2 Screenshot of command line (hw3_2)

```
54
98
3
1
30
```

Figure 3 p1_output.txt

```
K 3 5 9 A 10 2 8 4 Q 6 7 J
3 5 9 A 10 2 8 4 Q 6 7 J
5 9 A 10 2 8 4 Q 6 7 J 3
9 A 10 2 8 4 Q 6 7 J 3 5
A 10 2 8 4 Q 6 7 J 3 5 9
10 2 8 4 Q 6 7 J 3 5 9 A
2 8 4 Q 6 7 J 3 5 9 A 10
8 4 Q 6 7 J 3 5 9 A 10 2
4 Q 6 7 J 3 5 9 A 10 2 8
Q 6 7 J 3 5 9 A 10 2 8 4
6 7 J 3 5 9 A 10 2 8 4
7 J 3 5 9 A 10 2 8 4 6
J 3 5 9 A 10 2 8 4 6 7
3 5 9 A 10 2 8 4 6 7
5 9 A 10 2 8 4 6 7 3
9 A 10 2 8 4 6 7 3 5
A 10 2 8 4 6 7 3 5 9
10 2 8 4 6 7 3 5 9 A
2 8 4 6 7 3 5 9 A
8 4 6 7 3 5 9 A 2
4 6 7 3 5 9 A 2 8
6 7 3 5 9 A 2 8 4
7 3 5 9 A 2 8 4 6
3 5 9 A 2 8 4 6 7
5 9 A 2 8 4 6 7 3
9 A 2 8 4 6 7 3 5
A 2 8 4 6 7 3 5
2 8 4 6 7 3 5 A
8 4 6 7 3 5 A 2
4 6 7 3 5 A 2
6 7 3 5 A 2 4
7 3 5 A 2 4 6
3 5 A 2 4 6
5 A 2 4 6 3
A 2 4 6 3 5
2 4 6 3 5 A
4 6 3 5 A 2
6 3 5 A 2 4
3 5 A 2 4
5 A 2 4 3
A 2 4 3
2 4 3 A
4 3 A 2
3 A 2
A 2
2 A
A
```

Figure 4 p2_output.txt

○ Program Architecture

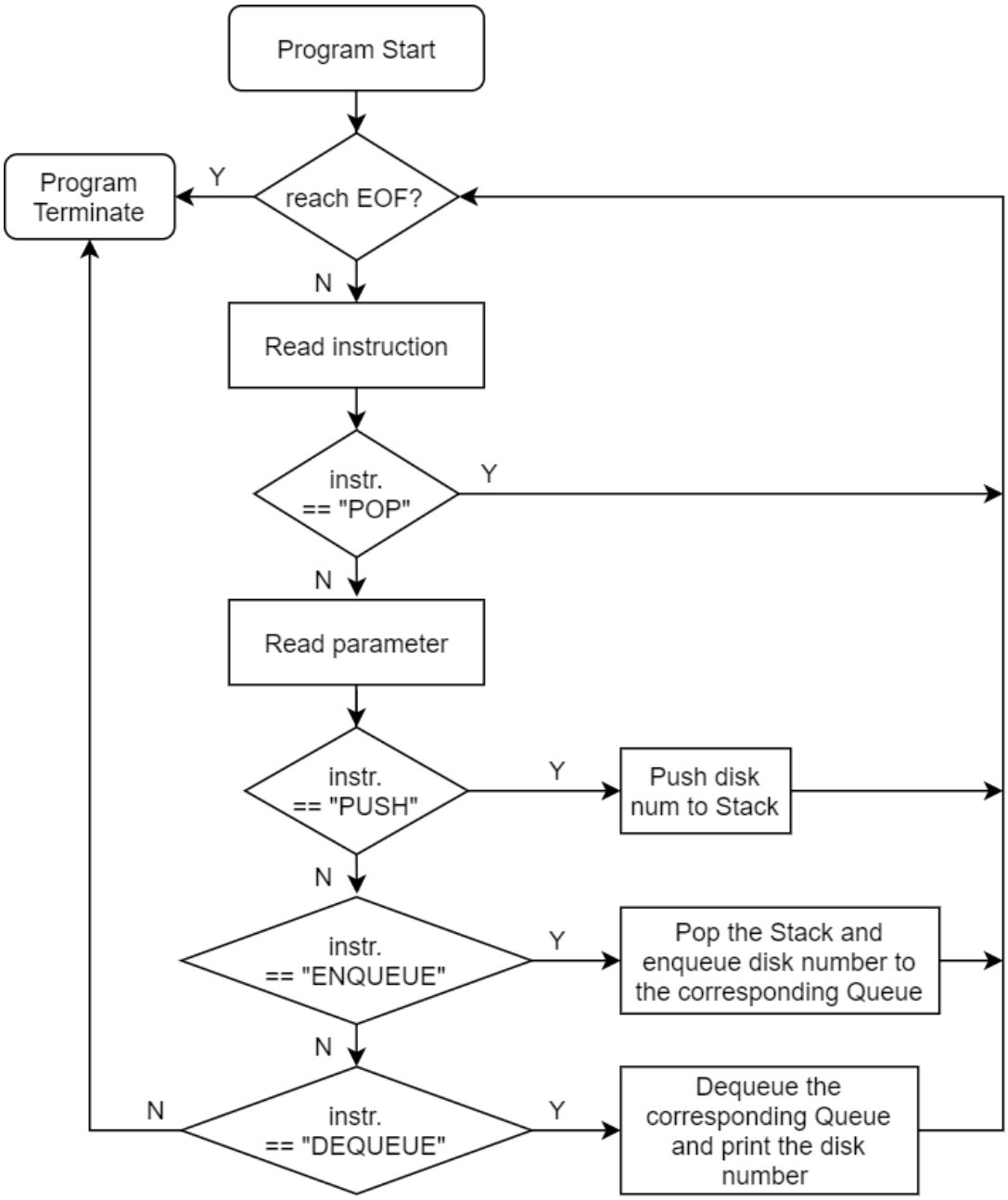

Figure 5 Flow chart of hw3_1

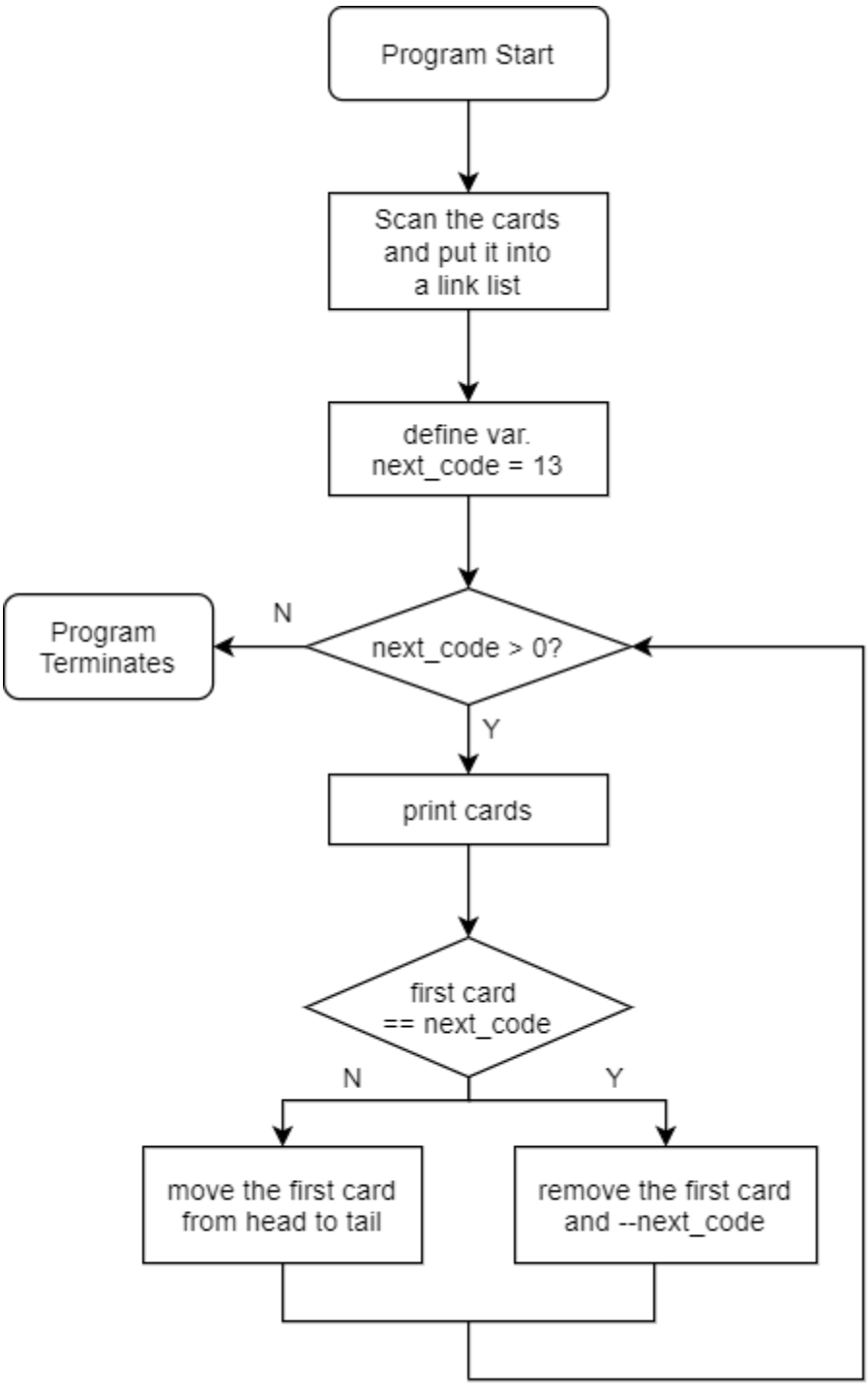

Figure 6 Flow chart of hw3_2

○ Program Functions

Double_LL.h

```
LinkList *create_ll();
```

Constructs a link list.

Parameters

None.

Return Value

Returns the new pointer of the link list.

- If construction fails, returns NULL.

```
Node *create_node(int val);
```

Constructs a node.

Parameters

`val`

The element that would be initialized in the constructed node.

Return Value

Returns the new pointer of the node.

- If construction fails, returns NULL.

```
void push_node(LinkList *lp, Node *np);
```

Inserts the node on the back of the link list.

Parameters

`lp`

The pointer of the link list.

`np`

The pointer of the node.

Return Value

None.

```
int pop_node(LinkList *lp);
```

Removes the node on the back of the link list.

📐 Parameters

`lp`

The pointer of the link list.

⮌ Return Value

Returns the back element before removal.

- If the link list is empty, program terminates.

```
void push_front_node(LinkList *lp, Node *np)
```

Inserts the node at the front of the link list.

📐 Parameters

`lp`

The pointer of the link list.

`np`

The pointer of the node.

⮌ Return Value

None.

```
int pop_front_node(LinkList *lp);
```

Removes the node at the front of the link list.

📐 Parameters

`lp`

The pointer of the link list.

⮌ Return Value

Returns the front element before removal.

- If the link list is empty, program terminates.

```
void free_LL(LinkList *lp);
```

Free all nodes in the link list.

📐 Parameters

`lp`

The pointer of the link list.

⮌ Return Value

None.

○ Program Design

本作業承接上個作業的精神—物件導向來實作，為了實作 doubly link list，我使用兩個 struct 來實現，一個是 `struct Node` 另一個是 `struct LinkList`，其宣告細節如下所示：

```
typedef struct Node
{
    struct Node *next;
    struct Node *prev;
    int val;
} Node;

typedef struct LinkList
{
    Node *head;
    Node *tail;
} LinkList;
```

使用 doubly link list 的好處是，可以在任意節點往前或往後移動。

另外，我在 `LinkList` 中，另外定義了末端節點指標，如此便可以直接存取最後一個 Node，節省 push 以及 pop 的時間。

- **Operating System**
  Ubuntu 20.04.1 LTS (Focal Fossa)
- **Compiler**
  gcc (Ubuntu 9.3.0-10ubuntu2) 9.3.0
- **Compile**

```
make
```

  - ⚠Notice

`hw3_2.c` 會使用到`../code_1/Double_LL.h`，編譯`hw3_2.c`前，請確認`code_1`目錄存在！

- **Run**

hw3_1:

```
./hw3_1 < input.txt > output.txt
```

hw3_2:

```
./hw3_2 < input.txt > output.txt
```