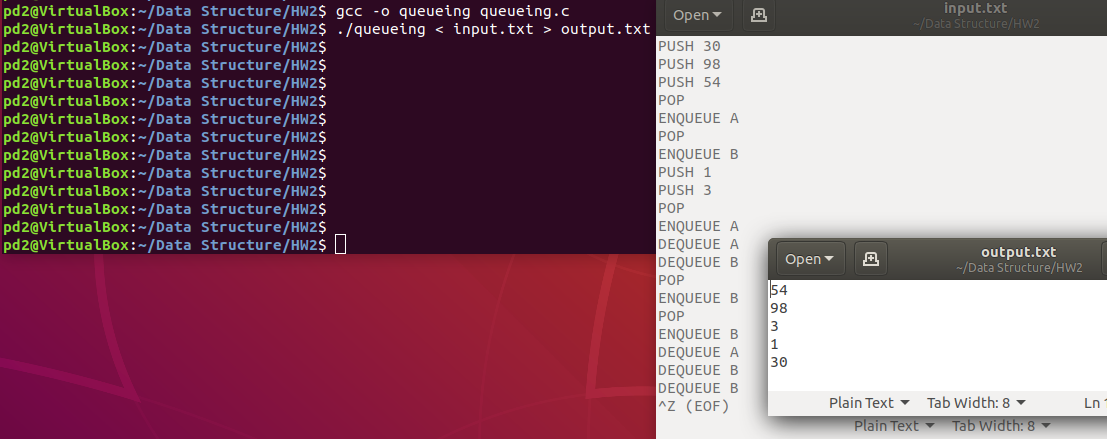
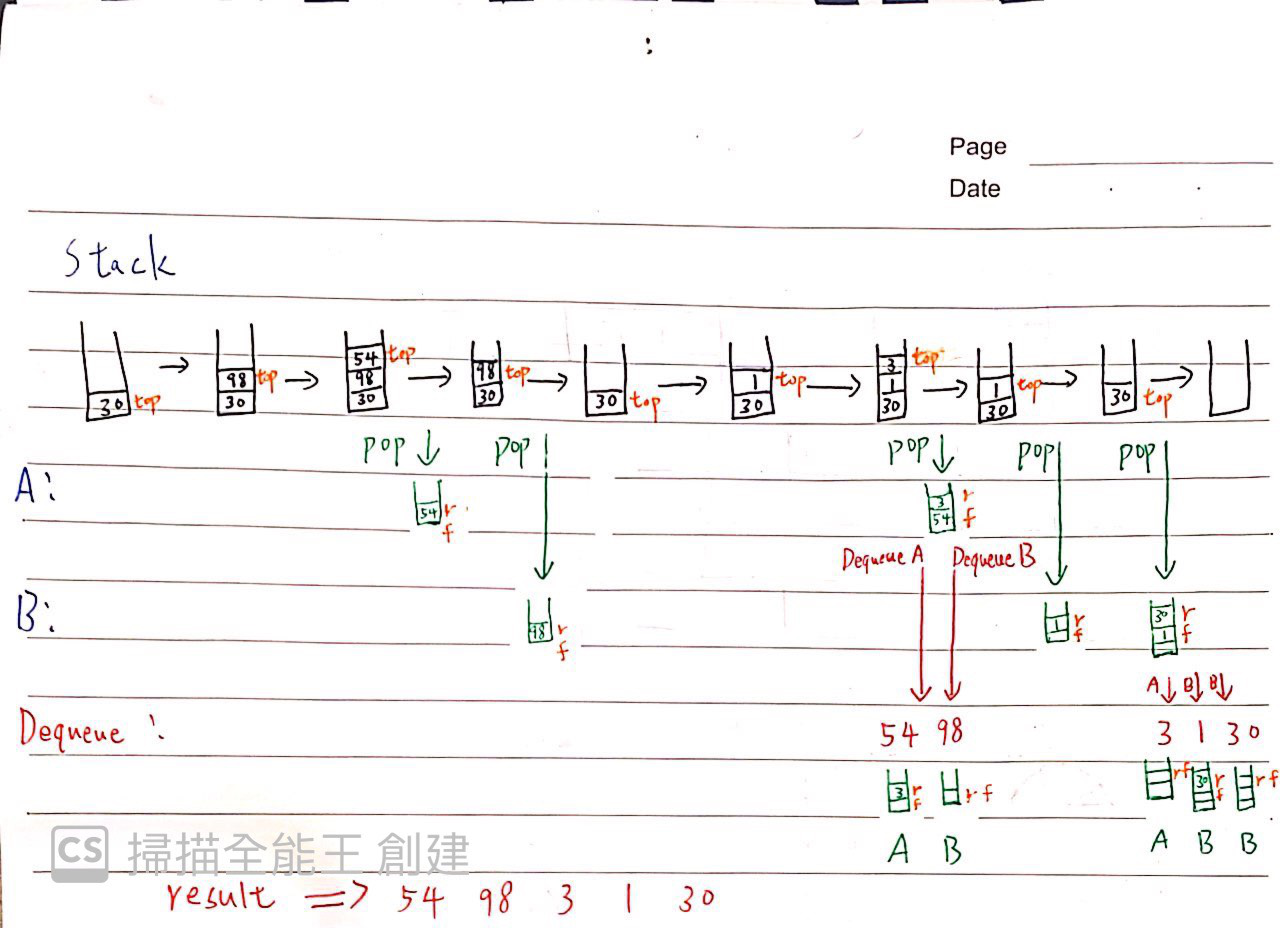
1. Result screenshot



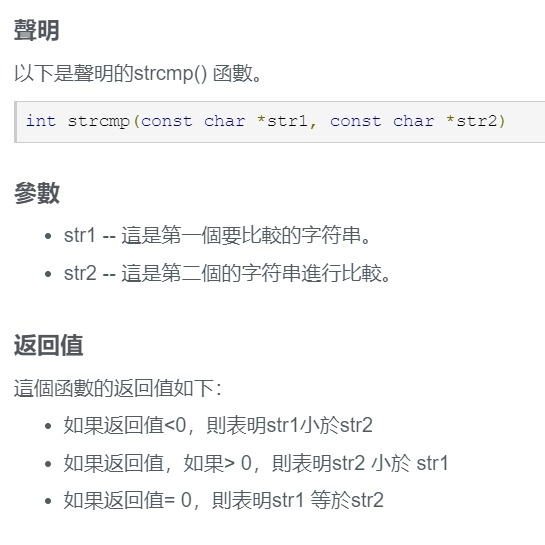
1. Program architecture



1. Program functions

(1)

strcmp():用來比對字串



(2)

void add\_plate(int top, int plate):用來把plate放入stack中

top – stack的頂端

plate – 要放入的plate的號碼

(3)

int delete\_plate(int top):用來取出stack中的plate

top – stack的頂端

return – 取出的plate的號碼

(4)

void enqueue(int queue[], int rear, int plate):把客人列入隊伍

queue[] – 要放入的隊伍(queue\_A或queue\_B)

rear – queue的頂端

plate – 客人手上拿的plate的號碼

(5)

int dequeue(int queue[], int front):用來取得離開隊伍的客人的plate的號碼

queue[] – 要離開的隊伍(queue\_A或queue\_B)

front – queue的底端

return – 客人放回的plate的號碼

1. How I design my program

設陣列plate\_stack來創立一個stack，並設立一個top\_plate來標示plate\_stack的頂端；設陣列queue\_A、queue\_B來建立兩個queue，並設立rear\_A、rear\_B來標示這兩個queue的頂端，front\_A、front\_B則標示這兩個queue的底端。

設一個陣列operation來接從input.txt接收到的字串，用strcmp加上if、else if，用類似switch case的方式來判斷現在是要PUSH、POP、ENQUEUE、或是DEQUEUE。

若接收到PUSH，則設一個int變數N來接收後面的號碼，也就是plate的號碼，並且呼叫add\_plate()這個函式，將N放入plate\_stack中，並將top\_plate+1。

若接收到POP則直接呼叫continue，因為POP後面是接ENQUEUE A或ENQUEUE B，也就是將plate\_stack中的N pop進隊伍中，而這一步我直接放到ENQUEUE的地方做，因此POP這邊我直接continue。

若接收到ENQUEUE，先設立一個陣列AorB，並把ENQUEUE後面的字串放入其中，也就是A或是B，並一樣透過strcmp()來判斷是A或B，接著assign rear\_A或rear\_B為其與MAX\_SIZE的餘數+1，這邊的用意是為了節省queue的空間(講義上圓圈的作法)，然後呼叫enqueue()來將plate\_stack中位於top\_plate的N放入queue\_A或queue\_B中，並且將top\_plate-1。

若接收到DEQUEUE，做法和ENQUEUE一樣，先判斷A或B，然後將assign front\_A或front\_B為其與MAX\_SIZE的餘數+1，然後呼叫dequeue()，將返回的結果直接輸出在output.txt中，並且將top\_plate-1。