利用 Stack 以及 dynamic memory allocation 完成下列程式:

1. 程式可以不斷讀取使用者輸入，並執行相關動作，直到使用者要求中止程式。
2. 產生兩個 stack，分別為 stack_big 及 stack_small；stack_big 用來儲存大寫英文字母(A-Z)，stack_small 用來儲存小寫英文字母(a-z)。
3. 當使用者輸入 大寫英文字母 A-Z 時，將該字母 push 進入 stack_big；當使用者輸入 小寫英文字母 a-z 時，將該字母 push 進入 stack_small。
4. 當使用者輸入 數字 0 時，從 stack_big 中 pop 出一個字元，並列印在螢幕上，若 stack_big 為"空"，則顯示 "stack big empty"。
5. 當使用者輸入 數字 1 時，從 stack_small 中 pop 出一個字元，並列印於螢幕上，若 stack_small 為"空"，則顯示"stack small empty"。
6. 當使用者輸入 數字 2 時，則依 字母順序[注 1] pop 出兩個 stack 的內容並列印於螢幕上，最後當兩個 stack 皆為 empty 後則 destroy 此兩個 stacks，然後結束程式。

注 1: 字母順序為 : A, a, B, b, C, c, D, d, ... Z, z

Example 1:
使用者輸入:
Tn Aa
P

small_stack

a
n

big_stack

P
A
T

Example 2:
使用者輸入:
Tn Aa
P → small_stack

a
n

big_stack

P
A
T

0
P (印出 P) → small_stack

a
n

big_stack

A
T

Dw

small_stack

w
a
n

big_stack

D
A
T

1
w (印出 w) →

small_stack

a
n

big_stack

D
A
T

2
aDAnT

small_stack

big_stack

結束

必要程式碼：

```
typedef struct _stack{
	char a;
	struct _stack *next;
}  STACK;

void push(char c, STACK ** list) {
	...
}

char pop(STACK ** list){
	...
}

int main (void) {
	STACK * stack_big = NULL;
	STACK * stack_small = NULL;
	... // 程式中所有列印結果必須在主程式中完成
	return 0;
}
```