Problem A
Priority Queue 實作

Time limit: 1 second
Memory limit: 512 megabytes

題目內容

實作一個 Priority Queue，Priority Queue 需可決定使用最大堆積或者最小堆積，需包含下列四種功能。

1. 往 Priority Queue 中插入數字，並且符合 Heap 的規則
2. 往 Priority Queue 中 pop 一個數字，並調整內部 Heap 使其符合規則，且需回傳該被刪除的數字
3. 查詢當前 Priority Queue 頂端的數字。
4. 回傳當前 Priority Queue 的大小。

此題達成方式比較多，總而言之需使用 Heap ，且在每次插入，刪除都爲 O(logn) 的時間複雜度即可。

建立兩個 Priority Queue **A** 以及 Priority Queue **B** 分別使用最小堆積以及最大堆積，每次會分別會對指定的 Priority Queue 進行操作，需在指定的條件下輸出。

輸入格式

第一行輸入一個數字 n 代表接下來有 n 個操作

接下來 n 行，有四種狀況

1. 輸入數字 1 *mode num* 中間分別以空白隔開，若 *mode* 爲 1 ，則在 Priority Queue A 中插入 *num* ，若 *mode* 爲 2 ，則在 Priority Queue B 中插入 *num*
2. 輸入 2 以及 *mode* 中間分別以空白隔開，若 *mode* 爲 1 ，則在 Priority Queue A 中進行 pop，若 *mode* 爲 1 ，則在 Priority Queue B 中進行 pop，若指定的 Priority Queue 爲空，不進行任何操作。
3. 輸入 3 以及 *mode* 中間分別以空白隔開，若 *mode* 爲 1 ，則輸出 Priority Queue A 中的最小值，若 *mode* 爲 2 ，則輸出 Priority Queue B 中的最大值，若指定的 Priority Queue 爲空，不進行任何操作。
4. 輸入 4 以及 *mode* 中間分別以空白隔開，若 *mode* 爲 1 ，則輸出 Priority Queue A 元素數量，若 *mode* 爲 2 ，則輸 Priority Queue B 中的元素數量，若指定的 Priority Queue 爲空，不進行任何操作。

輸出格式

在進行 2 3 4 操作時需要輸出

1. 進行操作 2 時，需輸出指定 Priority Queue 被 pop 掉的數字，若該 Priority Queue 爲空，輸出 -1
2. 進行操作 3 時，需輸出指定 Priority Queue 中的 heap 堆頂的數值，若該 Priority Queue 爲空，輸出 -1
3. 進行操作 4 時，需輸出指定 Priority Queue 的元素數量。

技術規格

- $1 \leq n \leq 10^6$
- $1 \leq num \leq 10^9$
- $1 \leq mode \leq 2$

範例輸入 1

```
6
1 1 8
1 2 7
1 2 9
1 1 3
2 1
4 2
3 1
4 1
```

範例輸出 1

```
3
2
8
1
```

範例輸入 2

```
11
1 1 4
1 2 10
2 1
2 1
1 1 7
1 2 3
1 2 8
3 2
4 2
4 1
3 1
```

範例輸出 2

```
4
-1
10
3
1
7
```

Hint

可以自己寫一個 compare function ，用傳入的參數當作比較基準，此方法在在完成兩種堆積的 Priority Queue 上可以省略很多程式碼

一開始 Priority Queue 中的 heap Array 可以直接開到題目規定的 10^6 量級，不過有興趣的同學可以上網查詢 Dynamic Array Amortized Analysis 去了解如何在時間已及空間內尋求平衡

此次題目可能較難，同學可以多參考課本，或者到 Github 以及其他網路資源尋求解答，也可以在課輔時間到場詢問助教，希望大家都能理解這邊的觀念。

Note

```
#define _CRT_SECURE_NO_WARNINGS // 第一行加這個，便可正常使用 scanf
struct priorityQueue{
    int size;
    int mode; // 用來記錄這個 Priority Queue 是由小到大或者由大到小
    int *heap;
}typedef priorityQueue


void push(priorityQueue* pq, int num){

}

int pop(priorityQueue* pq){

}

int top(priorityQueue* pq){

}

int size(priorityQueue* pq){

}
```