```
#ifndef PQ_lib
#define PQ_lib

struct HeapType ///Heap_t
{
    void *elemts; // element array
    int numElemts; // number of elements，可以用來得知陣列是滿的或是空的
};
typedef struct HeapType Heap_t;

struct PQ   ///PQ_t
{
    H_class pqClass;   ///辨別是MaxHeap 或 MinHeap
    Heap_t   heap;   ///heap 的頭，即 Priority 的頭
    int maxSize, eleSize;   ///maxSize：陣列最大可存放的大小。eleSize：使用者使用結構的大小。
    int (*cp)(void *eleA, void *eleB);//結構裡面可以有函式指標，用來比較使用者想比較的數值。
};
typedef struct PQ PQ_t;
```

以上為使用者可使用 Heap_t 、 PQ_t 的結構。

若使用此 library，需先呼叫此函式建立 Priority Queue，做 Priority Queue 的初始化

void createPQ(PQ_t *pq , H_class pqClass, int eleSize, int maxSize, int (*cp)(void *eleA, void *eleB));

傳入值：

第一個參數：型態為 PQ_t，Priority Queue 頭的位址。

第二個參數：型態為 H_class，若想建構 maxHeap、傳入 MAXHEAP；建構 minaxHeap、傳入 MINHEAP。

第三個參數：型態為 int，使用者使用基本元素的大小。

第四個參數：型態為 int，使用者想用多長的 priority queue 的陣列大小。

第五個參數：函式指標。

需使用者寫一個可以比較兩個 node 的函式並傳入，如：

```
int Grade(void *EA, void *EB)
{
    if( ((stu*)EA)->grade > ((stu*)EB)->grade )
    {
        return 1;
    }
    return 0;
}
```

（以上為欲比較學生成績的函式）(stu 為該程式所使用的結構)

而需要其傳回的結果符合以下幾點

(1)當 EA 的值 > EB 的值，回傳 1

(2)當 EA 的值 <= EB 的值，回傳 0

回傳值：無回傳值

可使用的功能：

（１）辨別此 Priority Queue 是不是空的

int IsEmpty(PQ_t *pq); /* return 0: not empty, 1: empty*/

傳入值：型態為 PQ_t，Priority Queue 頭的位址。

回傳值：空的、回傳１；不是空的、回傳０。

（２）刪除 BST 中，與第一個傳入參數相等的值

int IsFull(PQ_t *pq); /* return 0: not full, 1:full */

傳入值：型態為 PQ_t，Priority Queue 頭的位址。

回傳值：滿的、回傳１；不是滿的、回傳０。

（３）加入一個 element 到 PQ

int Enqueue(PQ_t *pq, void* eleA); /* add an element into PQ */

第一個傳入值：型態為 PQ_t，Priority Queue 頭的位址。

第二個傳入值：型態為 void，想要加入 PQ 的 element 的位址。

回傳值：型態為 int，加入成功，回傳１；加入失敗（Priority Queue 滿的狀態），回傳０。

（４）從 PQ 中刪除一個 element

void *Dequeue(PQ_t *pq); /*delete an element from PQ */

傳入值：型態為 PQ_t，Priority Queue 頭的位址。

回傳值：型態為 void 指標，該指標指向刪除掉的節點。

#endif