Priority Queue Library 使用說明書

- 事前準備（如何使用 Priority Queue Library ？）：

1. 匯入相關函式庫: #include "pq.h"。
2. 建立自己要存放資料的結構，範例如下圖顯示:

```
typedef struct myElement {
    char ID[10];
    int math;
    int eng;
} student_t;
```

3. 根據你的結構，定義清楚比較(compareMath)與打印(print)的函式，範例如下顯示:

✧ 比較函式(compareMath): 讓 PQ 函式庫知道你要拿甚麼資料進行比較。請務必在 element A > element B 時回傳 1，反之為回傳 -1，相等時回傳 0 。

```
int compareMath(void *elementA, void *elementB) {
    int mathA = ((student_t *)elementA)->math;
    int mathB = ((student_t *)elementB)->math;
    //printf("mathA=%d mathB=%d\n",mathA,mathB);
    if(mathA>mathB) {
        return 1;
    }else if(mathA<mathB){
        return -1;
    }
    return 0;
}
```

✧ 打印函式(print):此 PQ 函式庫是以陣列的資料結構去實現，因此請用 sizeof(你存放資料的結構) 的方式來移動位址，來取得你每一顆在 PQ 中的資料，並將其印出。

```
void print(PQ_t *pq) {
    student_t *temp;
    for (int i=0; i<pq->heap.numElements;i++){
        temp = (student_t *)(pq->heap.elements+i*sizeof(student_t));
        printf("index=%d, ID=%s,math=%d, eng=%d\n",
        i,temp->ID, temp->math, temp->eng);
    }
}
```

➢ 了解 Priority Queue (PQ) Library 與 功能使用說明 :

```
void createPQ(PQ_t *pq, H_class pqClass, int elementSize, int
 maxSize, int (*compare)(void* elementA, void *elementB));
```

◆ 函式說明: 初始化一個 pq 的結構，並設定其 PQ 的種類(pqClass)、元素大小(elementSize)、最多元素個數(maxSize) 與 比較函數(compare)。

◆ 輸入要求:

1. PQ_t *pq: 為要**初始化** PQ 結構的變數名稱，可由以下語法建立:
   `PQ_t maxPQ;  //建立一個 PQ_t 結構，並將其命名為 maxPQ`
2. H_class pqClass: 設定其**PQ 的種類**，輸入 MINHEAP or MAXHEAP 來設定此 PQ 資料結構的**存放方式**。
3. elementSize: 告訴 lib 你**每個元素(你要存放資料的結構)的大小**為多少，可以透過 sizeof()來取得大小。
4. maxSize: 輸入一個整數告訴 lib 你最多需要存放幾筆元素資料。
5. compare: 利用函數指標告訴 lib 要**如何**將元素新增到 PQ 中，也就是使用事前準備好的 compareMath 函式，讓 lib 知道要以哪一個數值(此範例使用 Math 值)進行比較與新增元素。

◆ 使用範例:

```
createPQ(&maxPQ, MINHEAP, sizeof(student_t), 100, compareMath);
```

```
int Enqueue(PQ_t *pq, void * elementA);
```

◆ 函式說明: 將元素(elementA)加入到已經初始化過後的 PQ 結構中。

◆ 輸入要求與使用範例:

1. PQ_t *pq: 給定已經**初始化**過後的 PQ 結構的**位置**
2. elementA: 要加入到 PQ 結構中的元素**位置**，多筆資料的建立與新增可參考以下語法:
   - 建立 3 比學生資料:

```
student_t node[3]={
        {"C120308001", 70, 100},
        {"B220406001", 60, 90},
```

```
                {"D120306001", 80, 95},
    };
```

- 利用 for 迴圈將上述資料新增到 PQ 結構中：

```
for(int i=0;i<6;i++){
    Enqueue(&maxPQ, &node[i]);
}
```

```
void * Dequeue(PQ_t *pq);
```

- 函式說明：根據 pqClass 的規則將元素(elementA)從 PQ 結構取出。
- 輸入要求：
  1. PQ_t *pq：給定想要取出資料元素的 PQ 結構的位置。
- 使用範例：

```
for(int i=0;i<3;i++){
    student_t *temp = (student_t *)Dequeue(&maxPQ);
    printf("Dequeue: ID=%s, math=%d,eng=%d\n",temp->ID,
            temp->math,temp->eng);
    free(temp);
}
```

- 執行成果(以 MINHEAP 為例)：

```
Dequeue: ID=D220506001, math=10,eng=70
Dequeue: ID=B220406001, math=60,eng=90
Dequeue: ID=A220407001, math=65,eng=90
```

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

- 函式說明：判斷 PQ 結構是否為空或填滿。
- 輸入要求：
  1. PQ_t *pq：給定想要查看 PQ 結構的位置

根據上述說明完成 main.c 的程式碼後，可以透過以下語法使用 libpq.a 製作執行檔（main_static.exe)：

```
gcc main.c libBST.a  -o main_static
```

其中：main.c 是你的程式碼、是 libpq.a 是 PQ 的函式庫、main_static 是輸出的執行檔名稱。