Lab #8

Summer 2023

Requirements

In this optional lab, you will cover creating and maintaining a Priority Queue ADT. You are only given a partial definition of the PQ implementation, and you must derive the remainder of the implementation from the given complexity requirements. Note that you may need to define extra helper functions or struct types to complete this lab, and that these extra definitions must go in your **lab8.c** file.

In this lab you are given the following struct definitions:

```
// This is a partial definition, you must complete it in your lab8.c
   file

typedef struct Info Info;

typedef struct {
    Info *info;

} PQueue;
```

1.1 pqGetErrorCode

```
// O(1)
int  pqGetErrorCode(PQueue q);
```

Info: This function returns the error code from the most recently executed PQueue operation. 0 implies success, 1 implies out-of-memory error. Some functions may document additional error conditions. **NOTE: All queue functions assign an error code.**

1.2 pqInit

```
// O(1)
PQueue pqInit();
```

Info: This function returns an initialized PQueue variable. Every queue variable must be initialized before applying subsequent queue functions.

1.3 pqInsert

```
// O(n)

int pqInsert(void *data, int priority, PQueue q);
```

Info: This function enqueues an object and its associated priority (int) into the PQueue. For convenience, error code is returned directly (and also can be obtained via **pqGetErrorCode**)

1.4 pqRemoveMax

```
// O(1)
void * pqRemoveMax(PQueue q);
```

Info: This function returns the user object with highest priority, after removing it from the queue. If two objects have the same priority, then they are returned in FIFO order. NULL is returned if PQueue is empty and error code is set to 2. **NOTE: User can check error code if null objects are permitted in the queue.**

1.5 pqPeek

```
// O(1)
void * pqPeek(PQueue q);
```

Info: This function returns the user object with the highest priority, without removing it from the Queue.

1.6 pqGetSize

```
// O(1)
int pqGetSize(PQueue q);
```

Info: This function returns the number of objects in the PQueue.

1.7 pqFree

```
// O(n)
void pqFree(PQueue q);
```

Info: This function uninitializes a queue and frees all memory associated with it. **NOTE: value of PQueue variable is undefined after this function is applied, i.e., it should not be used unless initialized again using queueInit**.

Submission Information

Submit this assignment by using the mucsmake command.

Use the following submit command on tc.rnet:

mucsmake <assignment> <filename>

For example:

mucsmake lab8 lab8.c

Rubric: 20 points

1. Write required pqGetError Code function
   * 2 points
2. Write required pqInit function
   * 3 points
3. Write required pqInsert function
   * 4 points
4. Write required pqRemoveMax function
   * 4 points
5. Write required pqPeek function
   * 3 points
6. Write required pqGetSize function
   * 2 points
7. Write required pqFree function
   * 2 points

Notice:

1. Do NOT change the given .h file or function prototype.
2. All of your lab submissions must compile under GCC using the *−Wall* and *−Werror* flags to be considered for a grade.
3. You are expected to provide proper documentation in every lab submission, in the form of code comments.
4. Make sure to reference the source of any code that was not created independently by yourself. For example, if you used code which was presented in class lectures, the source would be something like “CS 2050 Course Notes by Jim Ries”.