Write the implementation file, priority\_queue.c, for the interface in the given header file, priority\_queue.h. Turn in your priority\_queue.c file and a suitable main program, main.c, that tests the opaque object.

priority\_queue.h is attached as a file to this assignment but is also listed here for your convenience. Your implementation file should implement the priority queue using a heap data structure. Submissions that implement the priority queue without using a heap will not receive any credit.

#ifndef PRIORITY\_QUEUE\_H

#define PRIORITY\_QUEUE\_H

enum status { FAILURE, SUCCESS };

typedef enum status Status;

enum boolean { FALSE, TRUE };

typedef enum boolean Boolean;

typedef void\* PRIORITY\_QUEUE;

//Precondition: Creates an empty priority queue that can store integer data items

// with different integer priority. Higher

// integer values indicate higher priority in the queue. For example, consider the

// priority and the data value to be key-value pairs where the priority is the key

// and the data is the value. The queue could hold 21,10 and 35, 5 so that the

// first item to be removed from the queue would be the data value 5 because

// it has higher priority (35) than the data value 10 which only has (21).

//Postcondition: Returns the handle to an empty priority queue.

PRIORITY\_QUEUE priority\_queue\_init\_default(void);

//Precondition: hQueue is a handle to a valid priority queue opaque object.

// Higher priority\_level values indicate higher priority in the queue.

// data\_item is simply a value we are storing in the queue.

//Postcondition: returns SUCCESS if the item was successfully added to the queue

// and FAILURE otherwise.

Status priority\_queue\_insert(PRIORITY\_QUEUE hQueue, int priority\_level, int data\_item);

//Precondition: hQueue is a handle to a valid priority queue opaque object.

//Postcondition: returns SUCCESS if the highest priority item was removed from the queue

// and FAILURE if the queue was empty.

Status priority\_queue\_service(PRIORITY\_QUEUE hQueue);

//Precondition: hQueue is a handle to a valid priority queue opaque object.

//Postcondition: returns a copy of the data value for the

// highest priority item in the queue. Sets the variable at the address

// referred to in pStatus to SUCCESS if there is

// at least one item in the queue and FAILURE otherwise. If pStatus is

// passed in as NULL then the status value is ignored for this run of the

// function.

int priority\_queue\_front(PRIORITY\_QUEUE hQueue, Status\* pStatus);

//Precondition: hQueue is a handle to a valid priority queue opaque object.

//Postcondition: returns TRUE if the priority\_queue is empty and FALSE otherwise.

Boolean priority\_queue\_is\_empty(PRIORITY\_QUEUE hQueue);

//Precondition: phQueue is a pointer to the handle of a valid priority queue opaque object.

//Postcondition: The opaque object will be free'd from memory and the handle pointed to

// by phQueue will be set to NULL.

void priority\_queue\_destroy(PRIORITY\_QUEUE\* phQueue);

#endif