A Queue Class

Version 1b

The *QUEUE* class

The header file, *queue.h*, should look like:

```
#ifndef __QUEUE_INCLUDED__
#define __QUEUE_INCLUDED__

#include <stdio.h>

typedef struct queue QUEUE;

extern QUEUE *newQUEUE(void (*d)(void *,FILE *),void (*f)(void *));
extern void enqueue(QUEUE *items,void *value);
extern void *dequeue(QUEUE *items);
extern void *peekQUEUE(QUEUE *items);
extern int sizeQUEUE(QUEUE *items);
extern void displayQUEUE(QUEUE *items,FILE *);
extern void displayQUEUEdebug(QUEUE *items,FILE *);
extern void freeQUEUE(QUEUE *items);

#endif
```

The header file contains the function signatures of your public methods while the code module, *queue.c*, contains their implementations.

The only local includes that *queue.c* should have are *queue.h* and the header file of the underlying data structure on which the queue is based.

Method behavior

Here are some of the behaviors your methods should have. This listing is not exhaustive; you are expected, as a computer scientist, to complete the implementation in the best possible and most logical manner.

- *newQUEUE* - The constructor is passed functions that knows how to display and free the generic values stored in the queue.
- *enqueue* - The *enqueue* method runs in constant or amortized constant time. The value to be enqueued is stored in the underlying data structure.
- *dequeue* - The *dequeue* method runs in constant or amortized constant time. The value to be dequeued is removed in the underlying data structure.
- *peekQUEUE* - The peek method returns the value ready to come off the queue, but leaves the queue unchanged. It runs in constant time.
- *sizeQUEUE* - The size method returns the number of items stored in the queue. It runs in amortized constant time.
- *displayQUEUE* - This display method prints the items stored in the queue. If the integers 5, 6, 2, 9, and 1 are enqueued in the order given, the method would generate this output:

  ```
  <5,6,2,9,1>
  ```

  with no preceding or following whitespace. An empty queue displays as `<>`.
- *displayQUEUEdebug* - This visualizing method simply calls the debug method of the underlying data structure.
- *freeQUEUE* - This method frees the queue by freeing the underlying data structure and then freeing the queue object itself.

Assertions

Include the following assertions in your methods:

- *newQUEUE* - The memory allocated shall not be zero.
- *dequeue* - The size shall be greater than zero.
- *peekQUEUE* - The size shall be greater than zero.

Testing your QUEUE class

Modify the testing program found in the *sll class description* to work with doubly-linked lists. Make sure you add additional testing to make sure the time constraints of all methods are met.