A Stack Class

Version 1c

The *STACK* class

For your stack class, the header file, *stack.h*, should look like:

```
#ifndef __STACK_INCLUDED__
#define __STACK_INCLUDED__

#include <stdio.h>

typedef struct stack STACK;

extern STACK *newSTACK(void (*d)(void *,FILE *),void (*f)(void *));
extern void push(STACK *items,void *value);
extern void *pop(STACK *items);
extern void *peekSTACK(STACK *items);
extern int sizeSTACK(STACK *items);
extern void displaySTACK(STACK *items,FILE *);
extern void displaySTACKdebug(STACK *items,FILE *);
extern void freeSTACK(STACK *items);

#endif
```

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

The only local includes that *stack.c* should have are *stack.h* and the header file of the underlying data structure upon which the stack 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 manner.

- *newSTACK* - The constructor is passed functions that knows how to display and free the generic values stored in the queue.
- *push* - The *push* method runs in constant or amortized constant time. The value to be pushed is stored in the underlying data structure.
- *pop* - The *pop* method runs in constant or amortized constant time. The value to be popped is removed in the underlying data structure.
- *peekSTACK* - The peek method returns the value ready to come off the stack, but leaves the stack unchanged. It runs in constant time.
- *sizeSTACK* - The size method returns the number of items stored in the stack. It runs in amortized constant time.
- *displaySTACK* - The display method prints the items stored in the stack. If the integers 5, 6, 2, 9, and 1 are pushed in the order given, the method would generate this output:

  ```
  |1,9,2,6,5|
  ```

  with no preceding or following whitespace. An empty stack displays as `||`.
- *displaySTACKdebug* - This visualizing method simply calls the debug method of the underlying data structure.
- *freeSTACK* - This method frees the stack by freeing the underlying data structure and then freeing the stack object itself.

Assertions

Include the following assertions in your methods:

- *newSTACK* - The memory allocated shall not be zero.
- *pop* - The size shall be greater than zero.
- *peekSTACK* - The size shall be greater than zero.

Testing your STACK class

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