IC250 Lab 4

This assignment has two questions. The second question builds on top of the first. Both questions are compulsory. There is also an optional question in the end.

1. Implement a stack of characters using a linked list. A simple list ADT is provided. Provide the basic stack operations in `stack.h` and `stack.c`:

   ```c
   // create stack
   void create(stack *s);
   // push a char into stack
   void push(stack *s, char x);
   // pop the top of the stack
   char pop(stack *s);
   // return the top of stack, without popping
   char peek(stack *s);
   // is the stack empty?
   int isEmpty(stack *s);
   // return the size of the stack
   int getSize(stack *s);
   ```

   Provide a menu-based implementation of the stack operations in a program called `stackMain.c`. The code must check for stack underflow. A sample run is shown below:

   ```
   ./stack <ENTER>
   Stack created.

   Choose option: pUsh, pOp, pEek, iseMpty, getSize, eXit: U
   Enter element to be pushed: a

   Choose option: pUsh, pOp, pEek, iseMpty, getSize, eXit: U
   Enter element to be pushed: b

   Choose option: pUsh, pOp, pEek, iseMpty, getSize, eXit: U
   Enter element to be pushed: c

   Choose option: pUsh, pOp, pEek, iseMpty, getSize, eXit: S
   Size of stack: 3

   Choose option: pUsh, pOp, pEek, iseMpty, getSize, eXit: E
   Top of stack: c

   Choose option: pUsh, pOp, pEek, iseMpty, getSize, eXit: O
   Popped: c

   Choose option: pUsh, pOp, pEek, iseMpty, getSize, eXit: E
   Top of stack: b
   ```

```
Choose option: pUsh, pOp, pEek, iseMpty, getSize, eXit: O
Popped: b

Choose option: pUsh, pOp, pEek, iseMpty, getSize, eXit: O
Popped: a

Choose option: pUsh, pOp, pEek, iseMpty, getSize, eXit: O
Cannot pop. Stack empty!

Choose option: pUsh, pOp, pEek, iseMpty, getSize, eXit: X
```

2. Using the stack ADT created above, implement a queue using two stacks. You must NOT implement a queue using the regular code for queue. You must use two stacks. Provide a menu-based interface for the queue:

```
./queue <ENTER>
Queue created (using two stacks!!)

Choose option: Enqueue, Dequeue, Head-tail, iseMpty, getSize, eXit: E
Enter element to be enqueued: a
Stack1:[x x x] Stack2:[x x x]

Choose option: Enqueue, Dequeue, Head-tail, iseMpty, getSize, eXit: E
Enter element to be enqueued: b
Stack1:[x x x] Stack2:[x x x]

Choose option: Enqueue, Dequeue, Head-tail, iseMpty, getSize, eXit: E
Enter element to be enqueued: c
Stack1:[x x x] Stack2:[x x x]

Choose option: Enqueue, Dequeue, Head-tail, iseMpty, getSize, eXit: S
Size of queue: 3
Stack1:[x x x] Stack2:[x x x]

Choose option: Enqueue, Dequeue, Head-tail, iseMpty, getSize, eXit: H
Head of queue: a, tail of queue: c
Stack1:[x x x] Stack2:[x x x]

Choose option: Enqueue, Dequeue, Head-tail, iseMpty, getSize, eXit: D
Dequeued: a
Stack1:[x x x] Stack2:[x x x]

Choose option: Enqueue, Dequeue, Head-tail, iseMpty, getSize, eXit: H
Head of queue: b, tail of queue: c
Stack1:[x x x] Stack2:[x x x]

Choose option: Enqueue, Dequeue, Head-tail, iseMpty, getSize, eXit: D
Dequeued: b
Stack1:[x x x] Stack2:[x x x]

Choose option: Enqueue, Dequeue, Head-tail, iseMpty, getSize, eXit: X
```

Important: *At the end of every queue operation, the contents of the two stacks must be displayed. This is displayed in the sample run above, with dummy stack contents. You must, of course, print the correct contents of the stack.*

Optional:

(a) Implement the stack using an array. In this case, you must check for stack overflow and underflow.