CENG218 Labwork 6

Using the Stack, Queue and LinkedList codes presented in the lecture;

1. Queue: implement a recursive `void invert(Queue, Queue&)` function so that a second queue in reverse order is created. Write a program that reads a queue until -1 is entered and displays both the original and inverted queues on screen.

   ```
   Fill your queue: 6 4 9 1 2 7 -1
   Original queue: 6 4 9 1 2 7
   Inverted queue: 7 2 1 9 4 6
   ```

2. LinkedList: implement a recursive `bool isMember(Node*, int)` LinkedList member function that searches for an integer in the node structure and returns true or false appropriately. Because Node* is a private member and can not be accessed from outside, implement a second function `bool search(int)` that operates as a public interface to isMember() function. Write a program that reads a linked list until -1 is entered and then searches for an element in this list.

   ```
   Fill your linked list: 6 4 9 1 2 7 -1
   6->4->9->1->2->7->NULL
   Enter search term: 2
   2 is available in the linked list

   Fill your linked list: 6 4 9 1 2 7 -1
   6->4->9->1->2->7->NULL
   Enter search term: 8
   8 is not available in the linked list
   ```

3. Stack: implement a recursive `bool filterOut(string)` Stack member function that searches for either even or odd elements in the stack (according to the parameter specified by the user) and removes them from the stack while keeping the rest of the numbers in the stack in the same order they were previously in. Write a program that reads a stack until -1 is entered, asks the user what to remove, and displays the final stack after removing specified elements.

   ```
   Fill your stack: 4 9 3 6 7 8 1 2 5 -1
   Select "even" or "odd" to filter: even
   Even elements have been removed from stack,
   the remaining elements are: 5 1 7 3 9

   Fill your stack: 4 9 3 6 7 8 1 2 5 -1
   Select "even" or "odd" to filter: odd
   Odd elements have been removed from stack,
   the remaining elements are: 2 8 6 4
   ```