1. Write a method called `palindrome` that takes a parameter, `list1` of type `StringADT` that represents a doubly linked list, and if the characters in the nodes represent a palindrome, assigns the same linked list to the implicit parameter. If the nodes of the linked list do not represent a palindrome, the implicit parameter should be the null pointer. The method should use the following but no other methods:

```java
public class StringADT{
    class Dink{
        Dink left, right;
        char data;
    }
    private Dink head, tail;

    public void palindrome(StringADT list1){...
    }
}
```
2. Write a method called `isSorted` that takes a linked list with pointer `front` that is an instance variable and returns true if the `data` fields of the nodes of the list taken consecutively are in ascending order and false if they are not. Use the following class and the fact that the pointer `front` points to the front of the list. Assume that linked list has already been created.

```java
public class Problem2 {
    class Link {
        Link next;
        char data;
    }
    Link front;

    public boolean isSorted() {
        ...
        ...
    }
}
```
3. Write a method called `reverse` that takes a parameter, `list1` of class `StringADT`, and produces a linked list pointed to by the `head` field of the implicit parameter and that has the nodes of the first list in reverse order. The procedure should use the following but no other methods:

```java
public class StringADT {
    class Link {
        Link next;
        char data
    }
    private Link head;

    public void reverse(StringADT list1) {
        ...
    }

    private void push(char ch) {
        ...
    }

    Show how `list2` is used in the driver program.