Alexandria University
Faculty of Engineering
Computer and Systems Engineering Department

CS122: Data Structures I
Assigned: Tuesday, April 8th, 2013
Due: Saturday, April 19th, 2013

Programming Assignment 3
Introduction to Linked Lists

1 Objectives

1. Implement singly/doubly linked lists.
2. Get introduced to ADTs (Abstract Data Types)
3. Implement an application for linked lists.

2 Linked List Interface

```java
public interface MyLinkedList {
    public void add(int index, Object element);
    public void add(Object element);
    public Object get(int index);
    public void set(int index, Object element);
    public void clear();
    public boolean isEmpty();
    public void remove(int index);
    public int size();
    public MyLinkedList sublist(int fromIndex, int toIndex);
    public boolean contains(Object o);
}
```

Returns	Function signature	Description
void	add(int index, Object element)	Inserts the specified element at the specified position in the list.
void	add(Object element)	Inserts the specified element at the end of the list.
Object	get(int index)	Returns the element at the specified position in this list.
void	set(int index, Object element)	Replaces the element at the specified position in this list with the specified element.
void	clear()	Removes all of the elements from this list.
boolean	isEmpty()	Returns true if this list contains no elements.
void	remove(int index)	Removes the element at the specified position in this list.
int	size()	Returns the number of elements in this list.
MyLinkedList	sublist(int fromIndex, int toIndex)	Returns a view of the portion of this list between the specified fromIndex and toIndex, inclusively.
boolean	contains(Object o)	Returns true if this list contains an element with the same value as the specified element.

It is required to implement the above interface **twice**. Once using a **Singly** Linked List implementation and once using a **Doubly** Linked List implementation in two different classes.

3 Testing

Provide a simple main function to test both implementations. Feel free to provide any valid interface for testing. A traditional testing scenario might look like:

- Initialize the linked list to point to your implementation and add some nodes to the list.
- Test the correct insertion of nodes by calling the get() method for all possible indices. Do not forget to test the special case of invalid index parameter.
- Add one more node at an index in the middle of the list. Use the get() method to assure that the node is added at the correct index.
- Change one node to point to another element with a value different than the old one by calling the set() method. Use the get() method to test if the node is updated properly.

- Use sublist() to choose some elements of the list. Assure that the size of the sublist is correct and the elements are the desired ones.
- Remove one node from the list and assert that the size of the list has been decreased and that the node has been deleted properly.
- Test the contains() method by calling the method with two integers: one that is in the list and another that is not in the list.
- Clear the elements of the list. Assure that the list is Empty.

4 Application

You are required to design a linked allocation system to represent and manipulate polynomials. You should use one of the linked list classes you implemented in part (A).
Each term of the polynomial will be represented as a node, using it's coefficient and exponent. Assume that you have 3 available polynomial variables: A, B and C, that can be set by the user and one variable R that acts as an accumulator for the results of operations on other polynomials.
Create a user-friendly, menu-driven system that performs the following operations:

- Read in a polynomial and store it in variable A, B, or C.
- Output a polynomial using a form that clearly displays it.
- Add two polynomials and store the result in R.
- Subtract two polynomials and store the result in R.
- Multiply two polynomials and store the result in R.
- Evaluate a polynomial at some point, a, where a is a floating point constant. In other words, substitute by the given value in your polynomial. Display the result as a floating point.
- Clear a polynomial.

Note that: a polynomial whose value is cleared or initially unset cannot be involved in an operation.
Your program can look like the following:

```
Please choose an action
-----------------------
1- Set a polynomial variable
2- Print the value of a polynomial variable
```

```
3- Add two polynomials
4- Subtract two polynomials
5- Multiply two polynomials
6- Evaluate a polynomial at some point
7- Clear a polynomial variable
====================================================================
1
Insert the variable name: A, B or C
A
Insert the polynomial terms in the form:
(coeff1, exponent1), (coeff2, exponent2), ..
(1, 1), (1, 0)
Polynomial A is set
====================================================================
Please choose an action
1- Set a polynomial variable, ... etc
====================================================================
1
Insert the variable name: A, B or C
B
Insert the polynomial terms in the form:
(coeff1, exponent1), (coeff2, exponent2), ..
(1, 1), (-1, 0)
====================================================================
Please choose an action
1- Set a polynomial variable, ... etc
====================================================================
5
Insert first operand variable name: A, B or C
C
Variable not set
Insert the first operand variable name: A, B or C
A
Insert the second operand variable name: A, B or C
B
Result set in R: (1, 2), (-1, 0)
====================================================================
Please choose an action
1- Set a polynomial variable, ... etc
====================================================================
2
Insert the variable name: A, B, C or R
R Value in R: x^2 - 1
```

5 Notes

- Take into consideration that your implementation will be used later in the project, so it has to be fully functional, well documented and reusable. Try very hard to clean up your implementation. Remove all unused variables. Do not write redundant and repeated code.
- You may use Checkstyle http://checkstyle.sourceforge.net/ with your IDE to ensure that your code style follows the JAVA coding style standards.
- You should work individually.
- Late submission is accepted for only one week.

Good Luck