LINKED LISTS WORKSHEET

Given the following Java code fragment:

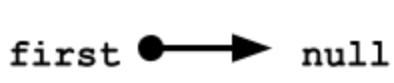
```
public class SomeClass {
      // Node type for linked list
      private class Node {
        private String item;
        private Node   next;
        private Node (String value) { item = value; }
      }
      // first must always refer to first Node in the linked list
      private Node first = null;

}
```

For each operation:

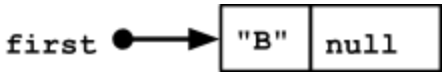
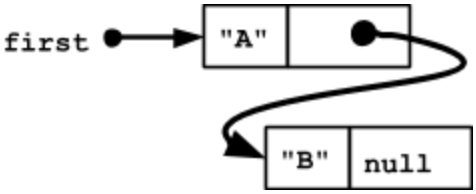
• Assume that `first` refers to beginning of the link list
• Draw a picture of the linked list before the operation
• Write the code to implement the operation
  ○ Note: `first` must always refer to first `Node` in the linked list
• Draw a picture of the linked list after the Java code is executed
	• Assume that `Node` is a nested class and its instance variables can be accessed directly - see Lecture 15 & pp. 575, 594
	• E.g.:
	
	`Node x = new Node("Princeton");`
	`StdOut.println(x.item);`
	`x.next = new Node("Tigers");`

1. Insert a new Node with `item = "B"` into the linked list

Before	Java code	After
	`Node temp = new Node("B");` `first = temp;`	

2. Insert a new `Node` with `item = "A"` into beginning of the linked list from (1)

Before	Java code	After

	`Node temp = new Node("A");`
	`temp.next = first;`
	`first = temp;`
		

LINKED LISTS WORKSHEET

3. Insert a new `Node` with `item = "D"` at the end of the linked list from (2)		
Before	Java code	After

4. Insert a new `Node` with `item = "C"` after the `Node` with `item = "B"` in the list from (3)		
Before	Java code	After

5. Using a do-while loop, print all the items in the linked list from (4).	6. Suppose you have a linked list. The start of the list is stored in `Node first`. Write a loop to add a new `Node` with `item = "E"` to the end of linked list. The list may contain 0 or more items.

`if (first == null)`
`  StdOut.println("EMPTY LIST");`
 	
`else`
`  Node current = first;`
`  do {`
 	
`  while (  );`	