King Saud University
College of Computer and Information Sciences
Computer Science Department

CSC 212 **First Semester 1439-1440**

Tutorial #6

Problem 1:

Write a static method *replace* (user of Stack ADT) that takes as input a stack *originalStack* and two elements *oldElem* and *newElem*. The method replaces all the occurrences of the element *oldElem* in *originalStack* with *newElem*.

Method: *public static<T> void replace (Stack<T> originalStack, T oldElem, T newElem)*

Example: assuming *originalStack* (top-to-bottom): 5, 7, 5, 3, 2. After calling *replace(originalStack, 5, 0)* then *originalStack* will be: 0, 7, 0, 3, 2.

```
public static<T> void replace (Stack<T> originalStack, T oldElem,
T newElem) {
	if (!originalStack.empty()) {
		Stack<T> tempStack = new LinkedStack<T>();
		T tempElem = null;
		while (!originalStack.empty()) {
			tempElem = originalStack.pop();
			if (tempElem.equals(oldElem))
				tempElem = newElem;
			tempStack.push(tempElem);
		}
		while (!tempStack.empty()) {
			originalStack.push(tempStack.pop());
	}
}
```

Problem 2:

Write a static method *insertAfter* (user of Stack ADT) that takes a non-full stack *originalStack*, an index *index*, and an element *newElem* as inputs. It should insert the element *newElem* after the element at position *index* in *originalStack*. You can assume *index* is within the range of the stack, and that the top element has an index of 0.

Method: *public static<T> void insertAfter(Stack<T> originalStack, int index, T newElem)*

Example: assuming *originalStack* (top-to-bottom): 5, 7, 5, 3, 2. After calling *insertAfter(originalStack, 2, 40)* then *originalStack* will be: 5, 7, 5, 40, 3, 2.

```
public static<T> void insertAfter(Stack<T> originalStack, int
index, T newElem) {
     Stack<T> tempStack = new LinkedStack<T>();
     int counter = 0;
     while (!originalStack.empty()) {
           tempStack.push(originalStack.pop(););
           if(counter == index)
                tempStack.push(newElem);
           counter++;
     }
     While (!tempStack.empty())
           originalStack.push(tempStack.pop());

}

//alternate solution
public static<T> void insertAfter(Stack<T> originalStack, int
index, T newElem) {
     Stack<T> tempStack = new LinkedStack<T>();
     for (int i = 0; i <= index; i++)
           tempStack.push(originalStack.pop());
     originalStack.push(newElem);
     while (!tempStack.empty())
           originalStack.push(tempStack.pop());

}
```

Problem 3:

Write the static method *removeBottom* (user of Stack ADT) that takes a stack *originalStack* as input, and removes the bottom-most element of it.

Method: *public static<T> void removeBottom(Stack<T> originalStack)*

Example: assuming *originalStack* (top-to-bottom): 5, 7, 5, 3, 2. After calling *removeBottom(originalStack)* the *stack* will be: 5, 7, 5, 3.

```
public static<T> void removeBottom(Stack<T> originalStack) {
	if (!originalStack.empty()) {
		Stack<T> tempStack = new LinkedStack<T>();
		while (!originalStack.empty())
			tempStack.push(originalStack.pop());
		tempStack.pop();
		while (!tempStack.empty())
			originalStack.push(tempStack.pop());
	}
}
```