Problem 1 solution:

```
public static<T> void replace (Stack<T> st, T x, T y) {
	Stack<T> temp = new Stack<T>();
	while(!st.empty()) {
		T e = st.pop();
		if(e.equals(x))
			e = y;
		temp.push(e);
	}
	while(!temp.empty()) {
		T e = temp.pop();
		st.push(e);
	}
}
```

Problem 2 solution:

```
public static<T> void insertAfter(Stack<T> st, int i, T e) {
	Stack<T> temp = new Stack<T>();
	int c = 0;
	while(!st.empty()) {
		T x = st.pop();
		temp.push(x);
		if(c == i)
			temp.push(e);
		c++;
	}
	while(!temp.empty()) {
		T x = temp.pop();
		st.push(x);
	}
}

//alternate solution credited to MT
public static<T> void insertAfter(Stack<T> st, int i, T e) {
	Stack<T> temp = new Stack<T>();
	for(int j = 0; j <= i; j++) {
		T x = st.pop();
		temp.push(x);
```

```
        }
        st.push(e);
        while(!temp.empty()) {
                T x = temp.pop();
                st.push(x);
        }
}
```

Problem 3 solution:

```
public static<T> void removeLast(Stack<T> st) {
        Stack<T> temp = new Stack<T>();
        while(!st.empty())
                temp.push(st.pop());
        if(!temp.empty())
                temp.pop();
        while(!temp.empty())
                st.push(temp.pop());
} f
```