1 More Practice with Linked Lists

```java
public class SLList {
  private class IntNode {
    public int item;
    public IntNode next;
    public IntNode(int item, IntNode next) {
      this.item = item;
      this.next = next;
    }
  }

  private IntNode first;

  public void addFirst(int x) {
    first = new IntNode(x, first);
  }
}
```

(a) Implement SLList.insert which takes in an integer x and an integer position. It inserts x at the given position. If position is after the end of the list, insert the new node at the end.

For example, if the SLList is 5 → 6 → 2, insert(10, 1) results in 5 → 10 → 6 → 2 and if the SLList is 5 → 6 → 2, insert(10, 7) results in 5 → 6 → 2 → 10. Additionally, for this problem assume that position is a non-negative integer.

```java
public void insert(int item, int position) {
}
```

(b) Add another method to SLList that recursively removes all nodes that contain a certain item. This method takes in an integer x and destructively changes

```java
public void remove(int x) {
```

}
the list.
For example, if the SLList is 3 \rightarrow 5 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 5, \texttt{removeItem(5)} results in 3 \rightarrow 4 \rightarrow 6.

```java
public void removeItem(int x) {
}
```

```java
private IntNode removeItemHelper(int x, IntNode current) {
}
```

(c) Extra: Add another method to the SLList class that reverses the elements. Do this using the existing IntNode objects (you should not use \texttt{new}).

```java
public void reverse() {
}
```
2 Arrays

(a) Consider a method that inserts an int item into an int[] arr at the given position. The method should return the resulting array. For example, if \( \text{arr} = [5, 9, 14, 15] \), \( \text{item} = 6 \), and \( \text{position} = 2 \), then the method should return \( [5, 9, 6, 14, 15] \). If position is past the end of the array, insert item at the end of the array. Assume we will only ever pass in a non-negative position.

Is it possible to write a version of this method that returns void and changes arr in place (i.e., destructively)? Hint: These arrays are filled meaning an array containing \( n \) elements will have length \( n \).

Fill in the below according to the method signature:

```java
public static int[] insert(int[] arr, int item, int position) {
}
```

(b) Write a non-destructive method replicate(int[] arr) that replaces the number at index \( i \) with \( \text{arr}[i] \) copies of itself. For example, \( \text{replicate([3, 2, 1])} \) would return \( [3, 3, 3, 2, 2, 1] \). For this question assume that all elements of the array are positive.

```java
public static int[] replicate(int[] arr) {
}
```