1. (12 marks) Consider the following structure of linked IntNode-objects:

![Diagram of linked IntNode-objects with nodes A, B, and C, and links between them.]

Execute the following piece of Java code relative to this structure and at the end of this execution, give diagrams of the linked lists of nodes accessible from the A, B, and C references, respectively.

```java
IntNode A, B, C, temp;

temp = A.getLink();
C.getLink().setLink(temp);
A.setLink(temp.getLink().getLink());
C.getLink().getLink().getLink().setLink(null);
temp.getLink().setLink(null);
```
Answer:

The state of the structure above after the execution of the given code is as follows:

```
A -> 1 -> 4 -> 5 -> 6 -> 2 -> 3
B -> 3
C -> 5 -> 6 -> 2 -> 3
```

2. (14 marks) Write the body of Java method

```
public int[] interleave(int[] a1, int[] a2)
```

which, given two integer arrays a1 and a2, creates and returns the array resulting when the elements of a1 and a2 are interleaved, e.g., [a1[0], a2[0], a1[1], a2[1], a1[2], a2[2], ...]. If one the array is shorter than the other, the elements in the portion of the longer array left after interleaving with the shorter array are simply added on to the end of the interleaving, e.g., if a1 = [1, 2] and a2 = [4, 5, 6, 7], their interleaving is [1, 4, 2, 5, 6, 7].
Answer:

```java
public static int[] interleaves(int[] a1, int[] a2)
{
    int[] rem, answer = new int[a1.length + a2.length];
    int i, min, pos;

    if (a1.length < a2.length) {
        min = a1.length;
        rem = a2;
    } else {
        min = a2.length;
        rem = a1;
    }

    pos = 0;

    for (i = 0; i < min; i++) {
        answer[pos++] = a1[i];
        answer[pos++] = a2[i];
    }

    for (i = min; i < rem.length; i++)
        answer[pos++] = rem[i];

    return (answer);
}
```

3. (24 marks)

In this question, we will develop various methods associated with a Java class `IntMBag` which encodes bags of integers in which each stored integer-value is in the range 0 to `max` inclusive, where `max` is initialized at bag-object creation. Each such bag-object has the following instance variables:

```java
private int[] data;
private int max;
```

We will use an array-implementation like that in Assignment #2, i.e., the length of array `data` is exactly equal to the number of items stored in the bag.
a) **(10 marks)** Write the body of **Java** method

```java
public int[] getSpectrum()
```

which returns the spectrum of the invoking bag-object, where the spectrum associated with a bag-object with maximum-value \( max \) is a \((max + 1)\)-length integer array \( s \) such that \( s[i], 0 \leq i \leq max \), is the number of items in the bag with value \( i \). For example, given bags \( b_1 = \{0, 1, 3, 1\} \) with \( max = 3 \) and \( b_2 = \{0, 1, 0, 0, 3\} \) with \( max = 6 \), the spectra of \( b_1 \) and \( b_2 \) are \([1, 2, 0, 1]\) and \([3, 1, 0, 1, 0, 0]\), respectively.

**Answer:**

```java
public int[] getSpectrum(){
    int[] spectrum = new int[max + 1];
    int i;

    for (i = 0; i < max + 1; i++)
        spectrum[i] = 0;
    for (i = 0; i < data.length; i++)
        spectrum[data[i]]++;
    return(spectrum);
}
```

b) **(14 marks)** Write the body of **Java** method

```java
public boolean equalSpectra(IntMBag b)
```

which returns **true** if the spectra of the invoking bag-object and \( b \) are equal and **false** otherwise. Let \( s_1 \) and \( s_2 \) be two spectra with max-values \( max_1 \) and \( max_2 \), respectively; without loss of generality, assume that \( s_1 \) is the shorter of the two spectra if they are not of the same length, i.e., \( max_1 \leq max_2 \). Two such spectra \( s_1 \) and \( s_2 \) are equal if for all \( i, 0 \leq i \leq max, s_1[i] = s_2[i] \), and for all \( j, max_1 < j \leq max_2, s_2[j] = 0 \). For example, given spectra \( s_1 = [0, 2, 1], s_2 = [0, 2, 1, 0, 0] \), and \( s_3 = [0, 2] \), \( s_1 \) and \( s_2 \) are equal, \( s_1 \) and \( s_3 \) are not equal, and \( s_2 \) and \( s_3 \) are not equal.
Answer:

```java
public boolean equalSpectra(IntMBag b) {
    int[] s1, s2, rem;
    int i, min;

    s1 = this.getSpectrum();
    s2 = b.getSpectrum();

    if (s1.length < s2.length) {
        min = s1.length;
        rem = s2;
    } else {  
        min = s2.length;
        rem = s1;
    }

    for (i = 0; i < min; i++)
        if (s1[i] != s2[i])
            return(false);

    for (i = min; i < rem.length; i++)
        if (rem[i] != 0)
            return(false);

    return(true);
}
```