Before attempting these problems, carefully read the web page on loop invariants at

www.cs.uofs.edu/~mccloske/courses/cmps144/invariants_lec.html

(to which there is a link on the course web page).

1. You have a jar containing $k$ red and $m$ blue marbles, where $k > 0$ and $m > 0$ (i.e., there is at least one marble of each color in the jar). You also have a repository of (infinitely many) red marbles. Repeat the following (non-deterministic) procedure until at most one marble remains in the jar:

   Choose (any) two marbles from the jar;
   if both are blue {
     put one back into the jar;
     throw away the other one;
     put a (new) red marble into the jar;
   }
   else { // at least one of the chosen marbles is red
     put a red one back into the jar;
     throw away the other one (which is either red or blue);
   }

(a) Give a convincing argument that the procedure necessarily terminates with a single marble remaining in the jar.

**Hint:** It is not the case that the number of marbles in the jar necessarily decreases during each iteration. However, if you assign one weight to each red marble and a different weight to each blue marble, you can show that the total weight of the marbles in the jar decreases during each iteration. It is up to you to come up with appropriate weights. **End of hint.**

(b) Make the strongest statement you can regarding the color of the last marble remaining in the jar. Will it always be the same, regardless of the values of $k$ and $m$? Or will it be a function of $k$ and $m$? Either way, give a convincing argument, making use of the loop invariant concept.

If the color of the last marble remaining is unpredictable (i.e., for some values of $k$ and $m$ it is possible to finish with a red marble but also possible to finish with a blue marble), demonstrate this.
2. At the end of the aforementioned web page on loop invariants there appears a development of a program that solves the two-color version of the Dutch National Flag problem. The program was developed using a proposed loop invariant as a guide.

Here you are asked to develop an alternative solution to the same problem. As on the web page, your program is not to modify the array except by swapping array elements. (Assume that there is a method swap such that the effect of making the call swap(a, k, j) is to swap the values in a[k] and a[j].)

The precondition is that every element in the array a[0..N - 1] (we use N as an abbreviation for the length of a, which in Java is written a.length) qualifies as being either RED or BLUE (but not both). The postcondition is as indicated in this picture:

```
| 0 | k | N |
-----------------+-----------------+-----------------|
| a | all RED | all BLUE | |
-----------------+-----------------+-----------------|
```

In words, this says that every element in a[0..k - 1] is RED and every element in a[k..N - 1] is BLUE. More formally, we could express this in the language of predicate logic as follows:

$$(\forall i \mid 0 \leq i < k \,: \, a[i] \text{ is RED}) \wedge (\forall i \mid k \leq i < N \,: \, a[i] \text{ is BLUE})$$

The loop invariant of your program should be as indicated in this picture:

```
| 0 | k | m | N |
-----------------+-----------------+-----------------+-----------------|
| a | all RED | | all BLUE | |
-----------------+-----------------+-----------------+-----------------|
```

In words, this says that every element in a[0..k - 1] is RED and every element in a[m..N - 1] is BLUE. (Notice that we obtained the invariant by replacing k in the second conjunct of the postcondition by the “fresh” variable m.) More formally, we can express this by

$$(\forall i \mid 0 \leq i < k \,: \, a[i] \text{ is RED}) \wedge (\forall i \mid m \leq i < N \,: \, a[i] \text{ is BLUE})$$

Arrive at your solution by correctly replacing each question mark in the following with either an expression or a sequence of statements, whichever is appropriate. The initialization of k and m should establish the invariant by making the ?-region cover the entire array. Each iteration of the loop should decrease the length of the ?-region.

```java
k = ?; m = ?;
while ( ? )
{
    if (a[k] is RED) { ? }
    else /* a[k] is BLUE */ { ? }
}
```