1. Develop a solution for the Three-Color Dutch National Flag problem. (The two-color version of the problem was featured in Homework #1.) Given is an array \( a[ ] \) each element of which satisfies exactly one among the predicates \( isRed() \), \( isWhite() \), and \( isBlue() \). Using \( N \) as a shorthand for \( a.length \), the postcondition is as indicated by this picture:

\[
\begin{array}{cccc}
0 & w & b & N \\
\hline
\end{array}
\]

\[
\begin{array}{cccc}
a & all \text{ RED} & all \text{ WHITE} & all \text{ BLUE} & \&\& 0\leq w\leq b\leq N \\
\hline
\end{array}
\]

This says that \( 0 \leq w \leq b \leq N \) and, in words, that every element in \( a[0..w-1] \) satisfies \( isRed() \), every element in \( a[w..b-1] \) satisfies \( isWhite() \), and every element in \( a[b..N-1] \) satisfies \( isBlue() \).

The loop invariant of your solution must be as indicated in this picture:

\[
\begin{array}{cccc}
0 & r & w & b & N \\
\hline
\end{array}
\]

\[
\begin{array}{cccc}
a & all \text{ RED} & ? & all \text{ WHITE} & all \text{ BLUE} & \&\& 0\leq r\leq w\leq b\leq N \\
\hline
\end{array}
\]

This says that \( 0 \leq r \leq w \leq b \leq N \) and, in words, that every element in \( a[0..r-1] \) satisfies \( isRed() \), every element in \( a[w..b-1] \) satisfies \( isWhite() \), and every element in \( a[b..N-1] \) satisfies \( isBlue() \). (It says nothing about the values of elements in \( a[r..w-1] \).) Solve the problem by completing the program on the next page. Replace each underline with an expression and fill in the body of each of the three branches of the if statement. As in the previous homework, the only manner in which \( a[ ] \) is allowed to be modified is by swapping pairs of elements using the method \( swap() \): the invocation \( swap(a, i, j) \) causes the values in locations \( i \) and \( j \) of \( a[ ] \) to be swapped.
/* pre: every element in a[0..N-1] satisfies exactly one among *   isRed(), isWhite(), and isBlue() */

r = ______;  w = ______;  b = ______;

/* loop invariant: *
  * every element in a[0..r-1] is RED    &&
  * every element in a[w..b-1] is WHITE &&
  * every element in a[b..N-1] is BLUE  &&  0<=r<w<=b<=N */

while ( __________ ) {

    if (isRed(a[_____])) {

    }

    else if (isWhite(a[_____])) {

    }

    else // isBlue(a[_____]) {

    }

} /* post: every element in a[0..w-1] is RED    &&
   * every element in a[w..b-1] is WHITE &&
   * every element in a[b..N-1] is BLUE  &&  0<=w<=b<=N */
2. Given are an array \( a[0..N - 1] \) (we use \( N \) as an abbreviation for \( a.length \)) whose elements are in ascending order and a value which is sought, called \( x \).

Formally, the precondition is
\[
(\forall j \mid 0 \leq j < N - 1 : a[j] \leq a[j + 1])
\]

The “output variable” is \( k \); its final value should indicate the location of the “leftmost” element in \( a[\ ] \), if one exists, whose value is greater than or equal to \( x \). If there is no such element in \( a[\ ] \), the final value of \( k \) should be \( N \).

In the form of a picture, the postcondition is

\[
\begin{array}{ccc}
  & k & N \\
0 & \text{all are } < x & \text{all are } \geq x \\
\hline
a & \text{all are } < x & \text{all are } \geq x
\end{array}
\]

In words, this says that every element in \( a[0..k - 1] \) is less than \( x \) and that every element in \( a[k..N - 1] \) is greater than or equal to \( x \).

As a loop invariant, we adopt the one indicated by this picture:

\[
\begin{array}{ccc}
  & \text{low} & \text{high} & N \\
0 & \text{all are } < x & ? & \text{all are } \geq x \\
\hline
a & \text{all are } < x & ? & \text{all are } \geq x
\end{array}
\]

In words, this says that every element in \( a[0..\text{low} - 1] \) is less than \( x \) and that every element in \( a[\text{high} + 1..N - 1] \) is greater than or equal to \( x \). (In the program’s comments, we abbreviate these to \( a[0..\text{low} - 1] < x \) and \( a[\text{high} + 1..N - 1] \geq x \), respectively.) The invariant says nothing about the relationship between \( x \) and the values in \( a[\text{low}..\text{high}] \).

Solve the problem by completing the program on the next page. Replace each underline in the program with an appropriate expression (or, in one case, a relational operator (e.g., \(<\), \(\leq\), etc.)).

Here are a few hints.

The purpose of the loop is to reduce the length of the \( ? \)-region of the array to zero. Hence, the assignment to \( \text{low} \) inside the loop should advance it as far to the right as possible (while still preserving the truth of the invariant); likewise, the assignment to \( \text{high} \) should advance it as far to the left as possible.

Look at the picture of the loop invariant above and imagine what it would look like in the case that the \( ? \)-region has length zero. This should suggest not only an appropriate loop guard but also an appropriate right hand side for the assignment statement following the loop.
/* pre: a[] is in ascending order

int low = _____;  int high = _____;  // establish the invariant

/* loop invariant: a[0..low-1] < x  &&  a[high+1..N-1] >= x */

while ( ___________ ) {

    int mid = (low + high) / 2;  // assertion: low <= mid <= high

    if (a[mid] ___ x) {

        low = ___________;  // move low as far to right as possible
    }
    else {

        high = ___________;  // move high as far to left as possible
    }

}

int k = ___________;  

/* post: a[0..k-1] < x  &&  a[k..N-1] >= x */