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

www.cs.uofs.edu/~mccloske/courses/cms144/invariants_lec.html
(to which there is a link on the course web page).

1. Complete the following program so that it is consistent with its documentation. Recall that, in Java, the remainder obtained when dividing a by b is given by the expression a % b. Hence, to say that a is divisible by (or, equivalently, is a multiple of) b is to say that a % b = 0.

    /* pre: x > 0 & y > 0
    * post: value returned is the least common (positive) multiple of x and y
    */
    public int lcm(int x, int y) {
        int k = __________;

        /* loop invariant: k is a positive multiple of x such that
        * no smaller positive multiple of x is divisible by y.
        * 0r, to state it more technically: for some integer q > 0,
        * k = qx and none of x, 2x, 3x, ..., (q-1)x is divisible by y
        */
        while ( ___________________________ ) {
            }

        return _____________;
    }
2. Supply code necessary for making the method below satisfy its specification, as described by its pre- and post-conditions. The purpose of the method is to “compress” an integer array whose elements are in ascending order by removing any duplicate values. The method’s return value reports the number of distinct values in the array. For example, the invocation 
\texttt{aryCompress(junk)} should have the effect of transforming

\begin{verbatim}
  0 1 2 3 4 5 6 7 8 9 10 11
+--------------------------------------------------+
\texttt{junk} | -2| -2| -2| 0 | 3 | 5 | 5 | 8 | 8 | 8 | 8 | 13|
+--------------------------------------------------+
\end{verbatim}

into

\begin{verbatim}
  0 1 2 3 4 5 6 7 8 9 10 11
+--------------------------------------------------+
\texttt{junk} | -2| 0 | 3 | 5 | 8 | 13| - | - | - | - | - | -|
+--------------------------------------------------+
\end{verbatim}

and yielding the value 6. (The \texttt{'}-\texttt{'s in the array indicate that we don’t care what values appear there.)

\begin{verbatim}
/* pre: \texttt{b.length > 0} \& \texttt{values in } b[0..b.length-1] \texttt{are in ascending order}
 * post: The segment \texttt{b[0..k-1]} contains, in increasing order, one copy of each
 *       value originally occurring in \texttt{b[0..b.length-1]}, where \texttt{k} is the value
 *       returned by the method.
 */
public static int aryCompress(int[] b) {

    int k = 1;       int m = ________;

    /* loop invariant: \texttt{b[0..k-1]} contains, in increasing order, one copy of each
     * value originally occurring in \texttt{b[0..m-1]} \& \texttt{b[m..b.length-1]} is unchanged
     */
    /*
    */
    while ( _________________ ) {

    }

    return k;
}
\end{verbatim}