1. **Rotating a vector**
   You are required to rotate an \( n \)-element vector \( x \) left by \( i \) positions in time proportional to \( n \), and with as little extra space as possible. For example, the vector of characters
   \[
   VLOMGQXVKURGLTLISJRV
   \]
   when rotated left by \( i = 4 \), results in
   \[
   GQXVKURGLTLISJRVVLOM
   \]
   One might try to solve the problem by copying the first \( i \) elements of \( x \) into a temporary array, moving the remaining \( n - i \) elements left \( i \) places, and then copying the first \( i \) from the temporary array back to the last positions in \( x \). However, the \( i \) extra locations used by this scheme make it too space-expensive. For a different approach, we could define a function to rotate \( x \) left by one position (in time proportional to \( n \)) and call it \( i \) times, but that would be too time-expensive.

   To solve the problem within the resource bounds will apparently require a more complicated program. One successful approach is a delicate juggling act: move \( x[0] \) to the temporary \( t \), then move \( x[i] \) to \( x[0] \), \( x[2i] \) to \( x[i] \), and so on (taking all indices into \( x \) modulo \( n \)), until we come back to taking an element from \( x[0] \), at which point we instead take the element from \( t \) and stop the process.

   If that process didn’t move all the elements, then we start at \( x[1] \) and continue until we move all the elements.

   Write a program to implement the vector rotation as described. Make sure that it works with all the possible values of \( i \) and \( n \). You may have to think in terms of the greatest common divisor of \( i \) and \( n \) for some help.

2. **Swapping two segments of a vector**
   This is a different view of the above problem: rotating the vector \( x \) is really just swapping the two segments of the vector \( ab \) to be the vector \( ba \), where \( a \) represents the first \( i \) elements of \( x \). Suppose \( a \) is shorter than \( b \). Divide \( b \) into \( b_l \) and \( b_r \) so that \( b_r \) is the same length as \( a \). Swap \( a \) and \( b_r \) to transform \( ab b_r \) into \( b_r b a \). The sequence \( a \) is in its final place, so we can focus on swapping the two parts of \( b \). Since this new problem has the same form as the original, we can solve it recursively.

3. **Swapping two segments of a vector (again)**
   Let us view the problem as transforming the array \( ab \) into array \( ba \), but let us also assume that we have a function that reverses the elements in a specified portion of the array. Starting with \( ab \), we reverse \( a \) to get \( a' b \), reverse \( b \) to get \( a'b' \), and then reverse the whole thing to get \( (a'b')' \), which is exactly \( ba \). This results in the following code for rotation; the comments show the results when input string \( x \) is rotated left four elements.

   ```
   x = "VLOMGQXVKU"
i = 4
n = strlen ( x );
reverse ( 0, i-1 ) /* MOLVGQXVKU */
reverse ( i, n-1 ) /* MOLVUKQXYZQ */
reverse ( 0, n-1 ) /* GQXKVULOM */
   ```

   Write a program to achieve the vector rotation using this algorithm.