The loop invariant is that, after each iteration, $A[p : k-1]$ holds the elements of $L[0 : i-1]$ and $R[0 : j-1]$ in sorted order, and that $k = i + j + p$.

The loop of lines 12-18 terminates when $i = n_L$ or $j = n_R$. Before they start, $i = j = 0$ and during the loop, EITHER i is incremented, OR j is incremented. Thus, at the end of the loop, either $i = n_L$ OR $j = n_R$. (This assumes $p \leq q \leq r$, if this is not true, none of the loops will run, and no element gets modified, effectively merging two empty arrays to give another empty array.) Thus, at the end of the loop, $A[p : k-1]$ holds the elements of $L[0 : n_L - 1]$ and $R[0 : j-1]$ OR it holds the elements of $L[0 : i-1]$ and $R[0 : n_R - 1]$ in sorted order.

Thus, if the first case is true, the for loop of lines 24-27 will execute until $j = n_R$. Otherwise, the for loop of lines 20-24 will execute until $i = n_L$. In either case, the loop invariant of the loop in lines 12-18 will be maintained, and finally, $A[p : k-1]$ will hold the elements of $L[0 : n_L - 1]$ and $R[0 : n_R - 1]$ in sorted order, with $k = i + j + p$, proving the correctness of the MERGE algorithm.

(Note, this is assuming the loop invariant is correct. We haven't proven the loop invariant stated earlier, but it is trivial to prove, just go through the **Initialisation**, **Maintenance**, and **Termination** methodically.)