A collection of intervals along a real number line is given by the list of left ends \( L \) and the corresponding list of right ends \( R \). List \( L \) is sorted. The intervals might sometimes overlap, and sometimes leave gaps. Write a program to find the total length of the number line that is covered by these intervals.

One approach is to add up the intervals. Another is to add up the gaps. I'll try the first approach, and I'll try a solution of the form

\[
s := 0. \quad \text{for } i := 0..\#L \text{ do } s := s + \text{something} \quad \text{od}
\]

using invariant \( A_i \) defined informally as

\[ s = \text{length covered by the intervals represented by } L[0;i] \text{ and } R[0;i] \]

and the body of the loop considers one new interval from \( L_i \) to \( R_i \). This new interval may already be totally covered, partly covered, or not covered at all. For \( i > 0 \) we can determine which of these three cases we have by knowing which of the previous intervals extended farthest to the right. Suppose it was the interval from \( L_k \) to \( R_k \) for some \( k: 0..i \).

### Totally covered case

\[ L_k \leq L_i \leq R_i \leq R_k \]

Then \( A_i \Rightarrow A'(i+1) \Leftrightarrow \text{ok} \).

### Partly covered case

\[ L_k \leq L_i \leq R_k \leq R_i \]

We have \( L_k \leq L_i \) because \( k<i \) and list \( L \) is sorted. Then \( A_i \Rightarrow A'(i+1) \Leftrightarrow s := s + R_i - R_k \).

### Not covered at all case

\[ L_k \leq R_k \leq L_i \leq R_i \]

Again we have \( L_k \leq L_i \) because \( k<i \) and list \( L \) is sorted. Then \( A_i \Rightarrow A'(i+1) \Leftrightarrow s := s + R_i - L_i \).

All three cases can be expressed as

\[ s := s + (R_k)\uparrow(R_i) - (R_k)\uparrow(L_i) \]

To keep track of \( R_k \), introduce variable \( r \), and strengthen \( A_i \) with the conjunct

\[ r = \text{farthest right point so far} \]

Now formally, the problem is \( P \) where

\[ P \equiv s' = \sum j: 0..\#L \cdot \uparrow R[0;j+1] - (\uparrow R[0;j])\uparrow(L_j) \]

and \( A_i \) is defined as

\[ r = \uparrow R[0;i] \wedge s = \sum j: 0..i \cdot \uparrow R[0;j+1] - (\uparrow R[0;j])\uparrow(L_j) \]

The program is now

\[ P \iff s := 0. \quad r := -\infty. \quad A 0 \Rightarrow A'(#L) \]

\[ A 0 \Rightarrow A'(#L) \iff \text{for } i := 0..\#L \text{ do } 0..\#L \wedge A i \Rightarrow A'(i+1) \quad \text{od} \]

\[ i: 0..\#L \wedge A i \Rightarrow A'(i+1) \iff s := s + r\uparrow(R_i) - r\uparrow(L_i). \quad r := r\uparrow(R_i) \]