264 (edit distance) Given two lists, write a program to find the minimum number of item insertions, item deletions, and item replacements to change one list into the other.

After trying the question, scroll down to the solution.

§ Here is the standard solution, which uses **for**-loops (Subsection 5.2.3), but **for**-loops are never necessary. We will change list A into list B. Let D: [(#A+1) * [(#B+1) * *nat*]] be an array-valued variable whose final value will be such that $D'\ i\ j$ = (the edit distance from A[0;..i] to B[0;..j]) . So the final answer will be D' (#A) (#B) .

```
(0)    for i:= 0;..#A+1 do D i 0:= i od.
(1)    for j:= 1;..#B+1 do D 0 j:= j od.
(2)    for i:= 1;..#A+1 do
(3)        for j:= 1;..#B+1 do
(4)            D i j:=    (D (i–1) (j–1) + if A i = B j then 0 else 1 fi)
(5)                     ↓ (D (i–1) j + 1)
(6)                     ↓ (D i (j–1) + 1) od od
```

Line 0 says A[0;..i] can be changed to [*nil*] by i deletions.
Line 1 says [*nil*] can be changed to B[0;..j] by j insertions.
Lines 2 and 3 fill in the interior of D .
On line 4, if we can transform A[0;..i–1] to B[0;..j–1] in D (i–1) (j–1) steps, and if $A\ i = B\ j$, we have transformed A[0;..i] to B[0;..j] . But if $A\ i \ne B\ j$ then we need to replace $A\ i$ by $B\ j$ which takes 1 step.
On line 5, if we can transform A[0;..i–1] to B[0;..j] in D (i–1) j steps, then we can transform A[0;..i] to B[0;..j] by deleting $A\ i$.
On line 6, if we can transform A[0;..i] to B[0;..j–1] in $D\ i$ (j–1) steps, then we can transform A[0;..i] to B[0;..j] by appending $B\ j$.
The shortest way to transform A[0;..i] to B[0;..j] is the minimum of the three ways from lines 4, 5, and 6.

To prove the correctness of this solution, find invariants for the **for**-loops. Or eliminate the **for**-loops and write specifications as in Chapter 4.