```
1  ---------- MODULE OT ----------
   Specification of OT (Operational Transformation) functions. It consists of the basic OT functions
   for two operations and more general ones involving operation sequences.
7  EXTENDS Op, TLC
8  ------------------------------
   OT (Operational Transformation) functions.

   Naming convention: I for "Ins" and D for "Del".

   The left "Ins" lins transformed against the right "Ins" rins.
18 XformII(lins, rins) ≜
19     IF lins.pos < rins.pos
20      THEN lins
21      ELSE IF lins.pos > rins.pos
22               THEN [lins EXCEPT !.pos = @ + 1]
23               ELSE IF lins.ch = rins.ch
24                        THEN Nop
25                        ELSE IF lins.pr > rins.pr
26                                 THEN [lins EXCEPT !.pos = @ + 1]
27                                 ELSE lins

   The left "Ins" ins transformed against the right "Del" del.
32 XformID(ins, del) ≜
33     IF ins.pos ≤ del.pos
34      THEN ins
35      ELSE [ins EXCEPT !.pos = @ − 1]

   The left "Del" del transformed against the right "Ins" ins.
40 XformDI(del, ins) ≜
41     IF del.pos < ins.pos
42      THEN del
43      ELSE [del EXCEPT !.pos = @ + 1]

   The left "Del" ldel transformed against the right "Del" rdel.
48 XformDD(ldel, rdel) ≜
49     IF ldel.pos < rdel.pos
50      THEN ldel
51      ELSE IF ldel.pos > rdel.pos
52               THEN [ldel EXCEPT !.pos = @ − 1]
53               ELSE Nop
54 ------------------------------

   Transform the left operation lop against the right operation rop with appropriate OT function.
59 Xform(lop, rop) ≜
60     CASE lop = Nop ∨ rop = Nop → lop
61       □ lop.type = "Ins" ∧ rop.type = "Ins" → XformII(lop, rop)
```

```
        □ lop.type = "Ins" ∧ rop.type = "Del" → XformID(lop, rop)
        □ lop.type = "Del" ∧ rop.type = "Ins" → XformDI(lop, rop)
        □ lop.type = "Del" ∧ rop.type = "Del" → XformDD(lop, rop)
```

Generalized OT functions on operation sequences.

Iteratively/recursively transforms the operation op against an operation sequence ops.

```
RECURSIVE XformOpOps(_, _)
XformOpOps(op, ops) ≜
    IF ops = ⟨⟩
        THEN op
        ELSE XformOpOps(Xform(op, Head(ops)), Tail(ops))
```

Iteratively/recursively transforms the operation op against an operation sequence ops. Being different from $XformOpOps$, $XformOpOpsX$ maintains the intermediate transformed operation

```
RECURSIVE XformOpOpsX(_, _)
XformOpOpsX(op, ops) ≜
    IF ops = ⟨⟩
        THEN ⟨op⟩
        ELSE ⟨op⟩ ∘ XformOpOpsX(Xform(op, Head(ops)), Tail(ops))
```

Iteratively/recursively transforms the operation sequence ops against an operation op.

```
XformOpsOp(ops, op) ≜
    LET opX ≜ XformOpOpsX(op, ops)
    IN  [i ∈ 1 .. Len(ops) ↦ Xform(ops[i], opX[i])]
```

Iteratively/recursively transforms an operation sequence $ops1$ against another operation sequence $ops2$.

See also Definition 2.13 of the paper "Imine @ $TCS06$".

```
RECURSIVE XformOpsOps(_, _)
XformOpsOps(ops1, ops2) ≜
    IF ops2 = ⟨⟩
     THEN ops1
     ELSE XformOpsOps(XformOpsOp(ops1, Head(ops2)), Tail(ops2))
```

The $CP1$ (C for Convergence) property.

$TODO$: refactor the generation of $op1$ and $op2$.

```
CP1 ≜
    ∀ l ∈ List, op1 ∈ Op, op2 ∈ Op :
        ∧ PrintT(ToString(l) ∘ ", " ∘ ToString(op1) ∘ ", " ∘ ToString(op2))
            It is not allowed to delete elements from an empty list.
        ∧ ∨ (l = ⟨⟩ ∧ (op1.type = "Del" ∨ op2.type = "Del"))
            Priorities of these two insertions cannot be the same.
          ∨ (op1.type = "Ins" ∧ op2.type = "Ins" ∧ op1.pr = op2.pr)
```

```
124                 Position of an insertion cannot be too large.
125             ∨ (op1.type = "Ins" ∧ op1.pos > Len(l) + 1)
126             ∨ (op2.type = "Ins" ∧ op2.pos > Len(l) + 1)
127                 Position of a deletion cannot be too large.
128             ∨ (op1.type = "Del" ∧ op1.pos > Len(l))
129             ∨ (op2.type = "Del" ∧ op2.pos > Len(l))
130                 The CP1 itself.
131             ∨ ApplyOps(⟨op1, Xform(op2, op1)⟩, l) = ApplyOps(⟨op2, Xform(op1, op2)⟩, l)
```

The generalized $CP1$ (C for Convergence) property.

See also Theorem 2.14 of the paper "Imine @ $TCS06$".

FIXME: Generate legal operation sequences.

```
140 GCP1 ≜
141     ∀ l ∈ List, ops1 ∈ SeqMaxLen(Op, 1), ops2 ∈ SeqMaxLen(Op, 1) :
142         ∨ (Head(ops1).type = "Del" ∨ Head(ops2).type = "Del")
143         ∨ ApplyOps(ops1 ∘ XformOpsOps(ops2, ops1), l) =
144           ApplyOps(ops2 ∘ XformOpsOps(ops1, ops2), l)
145
```

* Modification History
* *Last* modified *Fri Jul* 06 15:14:55 *CST* 2018 by *hengxin*
* Created Sun *Jun* 24 15:57:48 *CST* 2018 by *hengxin*