|  |  |  |
| --- | --- | --- |
| Stack | Input | Table Lookup |
| E | (i nor false) and true or i nor not i and true xor i | **E -> TY** |
| TY |  | **T -> FX** |
| FXY |  | **F -> P** |
| PXY |  | **P -> ( E )** |
| ( E ) XY | (i nor false) and true or i nor not i and true xor i |  |
| E ) XY | i nor false) and true or i nor not i and true xor i | **E -> TY** |
| TY ) XY |  | **T -> FX** |
| FXY ) XY |  | **F -> P** |
| PXY ) XY |  | **P -> i** |
| i XY ) XY | i nor false) and true or i nor not i and true xor i |  |
| XY ) XY | nor false) and true or i nor not i and true xor i | **X ->** |
| Y ) XY |  | **Y -> nor TY** |
| nor TY ) TY | nor false) and true or i nor not i and true xor i |  |
| TY ) TY | false) and true or i nor not i and true xor i | **T -> FX** |
| FXY ) XY |  | **F -> P** |
| PXY ) XY |  | **P -> false** |
| false XY ) XY | false) and true or i nor not i and true xor i |  |
| XY ) XY | ) and true or i nor not i and true xor i | **X ->** |
| Y ) XY |  | **Y ->** |
| ) XY | ) and true or i nor not i and true xor i |  |
| XY | and true or i nor not i and true xor i | **X -> and T** |
| and TY | and true or i nor not i and true xor i |  |
| TY | true or i nor not i and true xor i | **T -> FX** |
| FXY |  | **F -> P** |
| PXY |  | **P -> true** |
| true XY | true or i nor not i and true xor i |  |
| XY | or i nor not i and true xor i | **X ->** |
| Y |  | **Y -> or TY** |
| or TY | or i nor not i and true xor i |  |
| TY | i nor not i and true xor i | **T -> FX** |
| FXY |  | **F -> P** |
| PXY |  | **P -> i** |
| i XY | i nor not i and true xor i |  |
| XY | nor not i and true xor i | **X ->** |
| Y |  | **Y -> nor TY** |
| nor TY | nor not i and true xor i |  |
| TY | not i and true xor i | **T -> FX** |
| FXY |  | **F -> not F** |
| not FXY | not i and true xor i |  |
| FXY | i and true xor i | **F -> P** |
| PXY |  | **P -> i** |
| i XY | i and true xor i |  |
| XY | and true xor i | **X -> and T** |
| and TY | and true xor i |  |
| TY | true xor i | **T -> FX** |
| FXY |  | **F -> P** |
| PXY |  | **P -> true** |
| true XY | true xor i |  |
| XY | xor i | **X ->** |
| Y |  | **Y -> xor TY** |
| xor TY | xor i |  |
| TY | i | **T -> FX** |
| FXY |  | **F -> P** |
| PXY |  | **P -> i** |
| i XY | i |  |
| XY | - | **X ->** |
| Y | - | **Y ->** |
| - | - |  |

proc E;

        T(); Y();

end;

proc Y;

    case Next\_Token of

        T\_or : Read(T\_or);

                T();

                Y();

Write (E -> E or T);

        T\_nor : Read(T\_nor);

                T();

                Y();

Write (E -> E nor T);

        T\_xor : Read(T\_xor);

                T();

                Y();

Write (E -> E xor T);

        T\_) : Write (E → T);

        otherwise   Error;

    end;

end;

proc T;

        F(); X();

end;

proc X;

    if Next Token = T\_and

    then

        Read(T\_and);

T();

Write (T -> F and T);

    if Next Token = T\_nand

    then

        Read(T\_nand);

T();

Write (T -> F nand T);

else Write (T -> F);

end;

proc F;

    case  Next Token of

        T\_true, T\_false, T\_i, T\_( : P();

Write (F -> P);

        T\_not: Read(T\_not);

F();

Write (F -> not F);

        otherwise Error;

    end;

end;

proc P;

    case Next\_Token of

        T\_true : Read(T\_true);

Write (P -> true);

        T\_false : Read(T\_false);

Write (P -> false);

        T\_i : Read(T\_i);

Write (P -> i);

        T\_( : Read(T\_( );

E();

Read(T\_) );

Write (P -> (E));

        otherwise   Error;

    end;

end;

P -> i

F -> P

T -> F

E -> T

P -> false

F -> P

T -> F

E -> E nor T

P -> (E)

F -> P

P -> true

F -> P

T -> F

T -> F and T

E -> T

P -> i

F -> P

T -> F

E -> E or T

P -> i

F -> P

F -> not F

P -> true

F -> P

T -> F

T -> F and T

E -> E nor T

P -> i

F -> P

T -> F

E -> E xor T

xor

nor

and

or

i

not

true

and

i

nor

true

i

false

i