

Translating Base 10 to Roman

1) transform([0], []).

2) transform([4], [i,v]).

3) transform([9], [i,x]).

4) transform([Digit], [i | Rest]) :- ibt(1, 3, Digit),
Newdigit is Digit-1,
transform([Newdigit], Rest).

5) transform([Digit], [v | Rest]) :- ibt(5, 8, Digit),
Newdigit is Digit-5,
transform([Newdigit], Rest).

6a) ibt(L, H, I) :- number(I), !, L =< I, I =< H.

6b) ibt(L, H, I) :- I is L.

6c) ibt(L, H, I) :- L1 is L+1, L1 =< H, ibt(L1, H, I).

?- transform([7], Roman).

4) transform([Digit], [i | Rest]) :- ibt(1, 3, Digit),
Newdigit is Digit-1,
transform([Newdigit], Rest).

Digit ↔ 7 Roman ↔ [i | Rest]

?- ibt(1, 3, 7), Newdigit is 7-1, transform(Newdigit, Rest).
6abc) ibt FAILS

?- transform([7], Roman).

5) transform([Digit], [v | Rest]) :- ibt(5, 8, Digit),
Newdigit is Digit-5,
transform([Newdigit], Rest).

Digit ↔ 7 Roman ↔ [v | Rest]

?- ibt(5, 8, 7), Newdigit is 7-5,
transform(Newdigit, Rest).

6abc), is) Newdigit ↔ 2

?- transform([2], Rest).

?- transform([2], Rest).

4) transform([Digit], [i | Rest]) :- ibt(1, 3, Digit),
Newdigit is Digit-1,
transform([Newdigit], Rest).

Digit₂ ↔ 2 Rest ↔ [i | Rest₂]

?- ibt(1,3,2), Newdigit₂ is 2-1,
transform([Newdigit₂], Rest₂).

6abc), is) Newdigit₂ ↔ 1

?- transform([1], Rest₂). 4) Digit₃ ↔ 1 Rest₂ ↔ [i | Rest₃]

?- ibt(1,3,1), Newdigit₃ is 1-1, transform([Newdigit₃], Rest₃).

6abc), is) Newdigit₃ ↔ 0

?- transform([0], Rest₃). 1) Rest₃ ↔ []

?- □ (empty goal list — success)

Roman was bound to [v | Rest]

Rest was bound to [i | Rest₂]

Rest₂ was bound to [i | Rest₃]

Rest₃ was bound to []

So Roman was ultimately bound to [v | [i | [i | []]]]

which Prolog automatically simplifies to the list [v,i,i].

7) `transform([0,Digit], Answer) :- transform([Digit], Answer).`

8) `transform([4,Digit], [x,l | Rest]) :- transform([Digit], Rest).`

9) `transform([9,Digit], [x,c | Rest]) :- transform([Digit], Rest).`

10) `transform([Digit1,Digit2], [x | Rest]) :- ibt(1, 3, Digit1),
Newdigit is Digit1-1,
transform([Newdigit,Digit2], Rest).`

11) `transform([Digit1,Digit2], [l | Rest]) :- ibt(5, 8, Digit1),
Newdigit is Digit1-5,
transform([Newdigit,Digit2], Rest).`

?- `transform (Base10, [l,i,x]).`

11) `Rest ↔ [i,x] Base10 ↔ [Digit1,Digit2]`

?- `ibt(5, 8, Digit1), Newdigit is Digit1-5,
transform([Newdigit,Digit2], [i,x]).`

`6abc), is) Digit1 ↔ 5 Newdigit ↔ 0`

?- `transform([0,Digit2], [i,x]).`

?- transform([0,Digit2], [i,x]).

7) transform([0,Digit], Answer) :- transform([Digit], Answer).

Digit \leftrightarrow Digit2 Answer \leftrightarrow [i,x]

?- transform([Digit2], [i,x]).

3) transform([9], [i,x]). Digit2 \leftrightarrow 9

?- \square (empty goal list — success)

Base10 was bound to [Digit1,Digit2]

Digit1 was bound to 5

Digit2 was bound to 9

So Base10 = [5,9]

?- transform([Digit1,7], [l,x | Rest]).

PROLOG responds with:

Digit1 = 6

Rest = [v,i,i]; (67 = lxxvii)

Digit1 = 7

Rest = [x,v,i,i]; (77 = lxxvii)

Digit1 = 8

Rest = [x,x,v,i,i]; (87 = lxxxvii)

no

?- transform([9,Digit2], Roman), length(Roman, 5).

PROLOG responds with:

Digit2 = 3

Roman = [x,c,i,i,i]; (93 = xciii)

Digit2 = 7

Roman = [x,c,v,i,i]; (97 = xcvii)

no