Brian — PS 18 — 2025-04-15 — Solution

EIWL3 Sections 41 and 42

Exercises from *EIWL3* Section 41

In[1]:=
```
(* 41.1 *) Cases[IntegerDigits[Array[#^2 &, 100]], {___, x_, y_, ___} /; x == y]
```

Out[1]=
```
{{1, 0, 0}, {1, 4, 4}, {2, 2, 5}, {4, 0, 0}, {4, 4, 1}, {9, 0, 0}, {1, 1, 5, 6},
 {1, 2, 2, 5}, {1, 4, 4, 4}, {1, 6, 0, 0}, {2, 1, 1, 6}, {2, 2, 0, 9},
 {2, 5, 0, 0}, {3, 3, 6, 4}, {3, 6, 0, 0}, {3, 8, 4, 4}, {4, 2, 2, 5},
 {4, 4, 8, 9}, {4, 9, 0, 0}, {5, 7, 7, 6}, {6, 4, 0, 0}, {6, 8, 8, 9},
 {7, 2, 2, 5}, {7, 7, 4, 4}, {8, 1, 0, 0}, {8, 8, 3, 6}, {1, 0, 0, 0, 0}}
```

In[2]:=
```
(* 41.2 *) StringJoin /@
 Cases[Array[Characters[RomanNumeral[#]] &, 100], {___, "L", ___, "I", ___, "X", ___}]
```

Out[2]=
```
{XLIX, LIX, LXIX, LXXIX, LXXXIX}
```

In[3]:=
```
(* 41.3 *) f[list_] := Equal[list, Reverse[list]]
```

In[4]:=
```
(* 41.4 *) (* So as not to fill up my document, I just took the first 20: *)
Take[Cases[Partition[TextWords[WikipediaData["alliteration"]], 2, 1],
  {x_, y_} /; Characters[x][[1]] == Characters[y][[1]], 20]
(* Notice that this solution is case-sensitive. One could also convert the *)
(* characters to lower-case and then test to get a case-insensitive test. *)
```

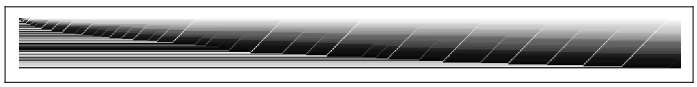
Out[4]=
```
{{or, of}, {as, a}, {Peter, Piper}, {pickled, peppers}, {Irish, It},
 {as, an}, {ideas, in}, {Icelandic, It}, {cartoon, characters},
 {the, term}, {identical, initial}, {several, special},
 {as, alliteration}, {stressed, syllables}, {as, an}, {lazy, languid},
 {languid, line}, {as, alliteration}, {be, because}, {such, syllables}}
```

In[5]:=
```
(* 41.5 *) Clear[x, y];
(* I was getting gibberish due to having defined x and y elsewhere. *)
FixedPointList[
  (# /. {x___, b_, a_, y___} /; b > a → {x, a, b, y}) &, {4, 5, 1, 3, 2}] // Grid
```

Out[6]=
```
4 5 1 3 2
4 1 5 3 2
1 4 5 3 2
1 4 3 5 2
1 3 4 5 2
1 3 4 2 5
1 3 2 4 5
1 2 3 4 5
1 2 3 4 5
```

In[7]:=
```
(* 41.6 *) Clear[x, y];
Transpose[FixedPointList[# /. {x___, b_, a_, y___} /; b > a → {x, a, b, y} &,
   RandomInteger[100, 50]]] // ArrayPlot
```

Out[8]=



In[9]:=
```
(* 41.7 *) FixedPoint[(# + 2 / #) / 2 &, 1.0]
(* This is a crafty way of computing Sqrt[2]. *)
```

Out[9]=
```
1.41421
```

In[10]:=
```
(* 41.8 *) FixedPointList[
 # /. {a_Integer, b_Integer} /; b ≠ 0 → {b, Mod[a, b]} &, {12 345, 54 321}]
```

Out[10]=
```
{{12 345, 54 321}, {54 321, 12 345}, {12 345, 4941},
 {4941, 2463}, {2463, 15}, {15, 3}, {3, 0}, {3, 0}}
```

In[11]:=
```
(* 41.9 *) FixedPointList[
 # /. {s[x_][y_][z_] → x[z][y[z]], k[x_][y_] → x} &, s[s][k][s[s[s]][s]][s]]
```

Out[11]=
```
{s[s][k][s[s[s]][s]][s], s[s[s[s]][s]][k[s[s[s]][s]]][s],
 s[s[s]][s][s][k[s[s[s]][s]][s]], s[s][s][s[s]][s[s[s]][s]],
 s[s[s]][s[s[s]]][s[s[s]][s]], s[s][s[s[s]][s]][s[s[s]][s[s[s]][s]]],
 s[s[s[s]][s[s[s]][s]]][s[s[s]][s][s[s[s]][s[s[s]][s]]]],
 s[s[s[s]][s[s[s]][s]]][s[s][s[s[s]][s[s[s]][s]]][s[s[s[s]][s[s[s]][s]]]]],
 s[s[s[s]][s[s[s]][s]]][
  s[s[s[s[s]][s[s[s]][s]]]][s[s[s]][s[s[s]][s]][s[s[s[s]][s[s[s]][s]]]]],
 s[s[s[s]][s[s[s]][s]]][s[s[s[s[s]][s[s[s]][s]]]][s[s][s[s[s[s]][s[s[s]][s]]]]][
    s[s[s]][s][s[s[s[s]][s[s[s]][s]]]]], s[s[s[s]][s[s[s]][s]]][
  s[s[s[s[s]][s[s[s]][s]]]][s[s[s[s]][s][s[s[s[s]][s[s[s]][s]]]]][
    s[s[s[s]][s[s[s]][s]]][s[s[s]][s][s[s[s[s]][s[s[s]][s]]]]]]]],
 s[s[s[s]][s[s[s]][s]]][s[s[s[s[s]][s[s[s]][s]]]][
   s[s[s][s[s[s[s]][s[s[s]][s]]]][s[s[s[s[s]][s[s[s]][s]]]]][s[s[s[s]][
       s[s[s]][s]]][s[s][s[s[s[s]][s[s[s]][s]]]][s[s[s[s[s]][s[s[s]][s]]]]]]]]],
 s[s[s[s]][s[s[s]][s]]][s[s[s[s[s]][s[s[s]][s]]]][s[s[s[s[s[s]][s[s[s]][s]]]]][
      s[s[s[s]][s[s[s]][s]]][s[s[s[s[s]][s[s[s]][s]]]]]]][
    s[s[s[s]][s[s[s]][s]]][s[s[s[s[s]][s[s[s]][s]]]]][
      s[s[s[s]][s[s[s]][s]]][s[s[s[s[s]][s[s[s]][s]]]]]]],
 s[s[s[s]][s[s[s]][s]]][s[s[s[s[s]][s[s[s]][s]]]][s[s[s[s[s[s]][s[s[s]][s]]]]][
      s[s[s[s]][s[s[s]][s]]][s[s[s[s[s]][s[s[s]][s]]]]]]][
    s[s[s[s]][s[s[s]][s]]][s[s[s[s[s]][s[s[s]][s]]]]][
      s[s[s[s]][s[s[s]][s]]][s[s[s[s[s]][s[s[s]][s]]]]]]]}
```

```
(* 41.10 *) IntegerDigits[Factorial[100]] /. {leading___, 0 ..} → {leading}
```

Out[37]=
```
{9, 3, 3, 2, 6, 2, 1, 5, 4, 4, 3, 9, 4, 4, 1, 5, 2, 6, 8, 1, 6, 9, 9, 2, 3, 8,
 8, 5, 6, 2, 6, 6, 7, 0, 0, 4, 9, 0, 7, 1, 5, 9, 6, 8, 2, 6, 4, 3, 8, 1, 6, 2, 1,
 4, 6, 8, 5, 9, 2, 9, 6, 3, 8, 9, 5, 2, 1, 7, 5, 9, 9, 9, 9, 3, 2, 2, 9, 9, 1, 5,
 6, 0, 8, 9, 4, 1, 4, 6, 3, 9, 7, 6, 1, 5, 6, 5, 1, 8, 2, 8, 6, 2, 5, 3, 6, 9, 7,
 9, 2, 0, 8, 2, 7, 2, 2, 3, 7, 5, 8, 2, 5, 1, 1, 8, 5, 2, 1, 0, 9, 1, 6, 8, 6, 4}
```

In[13]:=
```
(* 41.11 *) (* Below is one solution. Then I will give another. *)
Length /@ NestList[
  If[#[[1]] == 1, Join[Drop[#, 2], {0, 1}], Join[Drop[#, 2], {1, 0, 0}]] &, {1, 0}, 200]
```

Out[13]=
```
{2, 2, 3, 3, 4, 4, 5, 6, 6, 7, 8, 9, 9, 10, 11, 11, 12, 12, 13, 13, 14, 14, 15, 16, 16, 17,
 17, 18, 19, 19, 20, 21, 22, 22, 23, 23, 24, 24, 25, 25, 26, 26, 27, 28, 29, 29, 30,
 30, 31, 32, 32, 33, 33, 34, 35, 35, 36, 37, 37, 38, 38, 39, 40, 40, 41, 42, 43, 43,
 44, 44, 45, 45, 46, 46, 47, 47, 48, 48, 49, 50, 50, 51, 52, 53, 53, 54, 55, 55, 56,
 56, 57, 58, 58, 59, 59, 60, 61, 61, 62, 62, 63, 64, 64, 65, 66, 67, 67, 68, 69, 69,
 70, 70, 71, 71, 72, 72, 73, 74, 74, 75, 76, 77, 77, 78, 78, 79, 79, 80, 80, 81, 82,
 82, 83, 84, 85, 85, 86, 87, 87, 88, 88, 89, 89, 90, 90, 91, 92, 92, 93, 93, 94, 95,
 95, 96, 97, 98, 98, 99, 100, 100, 101, 101, 102, 103, 103, 104, 104, 105, 106,
 106, 107, 108, 109, 109, 110, 111, 111, 112, 112, 113, 113, 114, 114, 115, 116,
 116, 117, 117, 118, 119, 119, 120, 121, 122, 122, 123, 123, 124, 124, 125, 125}
```

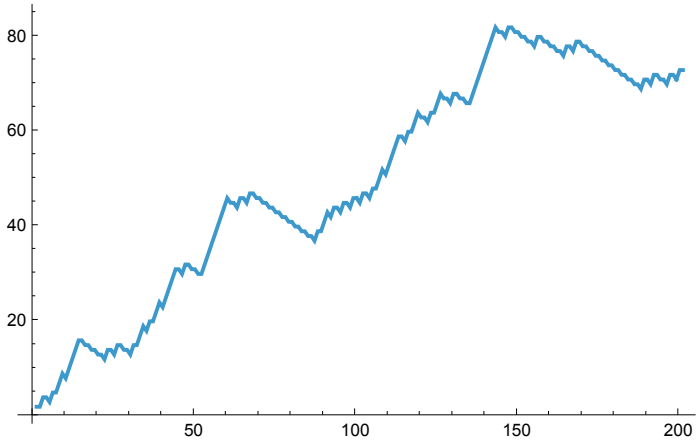
```
 (* Alternative 41.11 *)
(* A solution that is more in the spirit of this section of EIWL3: *)
changeroo[{1, y_, restOfList___}] := Join[{restOfList}, {0, 1}];
changeroo[{0, y_, restOfList___}] := Join[{restOfList}, {1, 0, 0}];
Length /@ NestList[changeroo, {1, 0}, 200]
```

Out[16]=
```
{2, 2, 3, 3, 4, 4, 5, 6, 6, 7, 8, 9, 9, 10, 11, 11, 12, 12, 13, 13, 14, 14, 15, 16, 16, 17,
 17, 18, 19, 19, 20, 21, 22, 22, 23, 23, 24, 24, 25, 25, 26, 26, 27, 28, 29, 29, 30,
 30, 31, 32, 32, 33, 33, 34, 35, 35, 36, 37, 37, 38, 38, 39, 40, 40, 41, 42, 43, 43,
 44, 44, 45, 45, 46, 46, 47, 47, 48, 48, 49, 50, 50, 51, 52, 53, 53, 54, 55, 55, 56,
 56, 57, 58, 58, 59, 59, 60, 61, 61, 62, 62, 63, 64, 64, 65, 66, 67, 67, 68, 69, 69,
 70, 70, 71, 71, 72, 72, 73, 74, 74, 75, 76, 77, 77, 78, 78, 79, 79, 80, 80, 81, 82,
 82, 83, 84, 85, 85, 86, 87, 87, 88, 88, 89, 89, 90, 90, 91, 92, 92, 93, 93, 94, 95,
 95, 96, 97, 98, 98, 99, 100, 100, 101, 101, 102, 103, 103, 104, 104, 105, 106,
 106, 107, 108, 109, 109, 110, 111, 111, 112, 112, 113, 113, 114, 114, 115, 116,
 116, 117, 117, 118, 119, 119, 120, 121, 122, 122, 123, 123, 124, 124, 125, 125}
```

In[17]:=
```
(* 41.12 *) (* Below is my first solution. Then I will give another. *)
switch[foo_] := Module[{fooDropped = Drop[foo, 2]}, Switch[foo[[1]], 0,
   Join[fooDropped, {2, 1}], 1, Append[fooDropped, 0], 2, Join[fooDropped, {0, 2, 1
     , 2}]]]
```

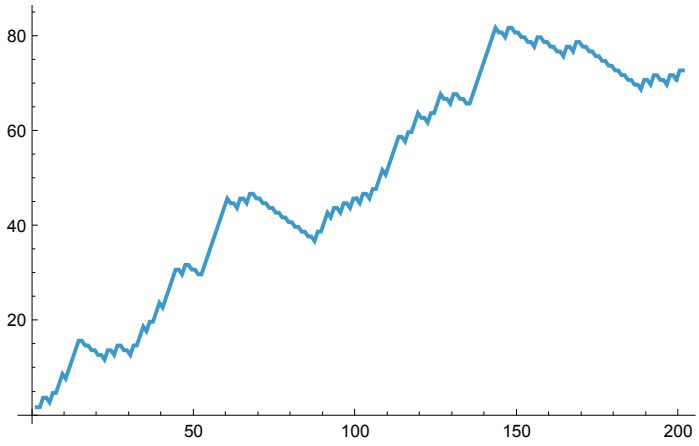
In[18]:=
```
ListLinePlot[Length /@ NestList[switch, {0, 0}, 200]]
```

Out[18]=




```
(* Alternative 41.12 *)
(* A solution that is more in the spirit of this section of EIWL3: *)
swaperoo[{0, _, foo___}] := Join[{foo}, {2, 1}];
swaperoo[{1, _, foo___}] := Join[{foo}, {0}];
swaperoo[{2, _, foo___}] := Join[{foo}, {0, 2, 1, 2}];
ListLinePlot[Length /@ NestList[swaperoo, {0, 0}, 200]]
```

Out[41]=



Exercises from *EIWL3* Section 42

In[24]:=
```
(* 42.1 *) StringReplace["1 2 3 4", " " → "---"]
```

Out[24]=
```
1---2---3---4
```

In[25]:=
```
(* Alternative 42.1 *) StringRiffle[StringSplit["1 2 3 4"], "---"]
```

Out[25]=
```
1---2---3---4
```

In[26]:=
```
(* 42.2 *)Sort[ToExpression /@ StringCases[WikipediaData["computers"],
   DigitCharacter ~~ DigitCharacter ~~ DigitCharacter ~~ DigitCharacter]]
```

Out[26]=
```
{1000, 1235, 1357, 1357, 1595, 1613, 1620, 1630, 1640, 1770, 1822, 1831, 1833,
 1835, 1872, 1872, 1876, 1876, 1888, 1890, 1897, 1901, 1901, 1906, 1914, 1920,
 1920, 1925, 1927, 1930, 1934, 1936, 1936, 1937, 1937, 1938, 1939, 1940, 1941,
 1941, 1942, 1943, 1943, 1943, 1943, 1944, 1945, 1945, 1945, 1945, 1945, 1945,
 1947, 1947, 1947, 1948, 1948, 1949, 1950, 1950, 1950, 1950, 1950, 1951,
 1951, 1952, 1953, 1953, 1955, 1955, 1955, 1955, 1957, 1958, 1958, 1959,
 1959, 1960, 1962, 1964, 1967, 1968, 1970, 1970, 1970, 1970, 1990, 1998,
 2000, 2000, 2000, 2016, 2400, 2468, 4000, 4004, 5000, 5100, 6502, 6510}
```

In[27]:=
```
(* 42.3 *)
StringCases[WikipediaData["computers"], "=== " ~~ Shortest[x___] ~~ " ===" → x]
```

Out[27]=
```
{Pre-20th century, First computer, Electromechanical calculating machine,
 Analog computers, Digital computers, Electromechanical,
 Vacuum tubes and digital electronic circuits, Modern computers,
 Concept of modern computer, Stored programs, Transistors, Integrated circuits,
 Mobile computers, By architecture, By size, form-factor and purpose,
 History of computing hardware, Other hardware topics, Input devices,
 Output devices, Control unit, Central processing unit (CPU),
 Arithmetic logic unit (ALU), Memory, Input/output (I/O), Multitasking,
 Multiprocessing, Languages, Programs, Stored program architecture,
 Machine code, Programming language, Low-level languages, High-level languages,
 Program design, Bugs, Computer architecture paradigms, Artificial intelligence}
```

In[28]:=
```
(* 42.4 *)
Apply[StringTemplate["``+``=``"], Array[{#1, #2, #1 + #2} &, {9, 9}], {2}] // Grid
```

Out[28]=

1+1=2	1+2=3	1+3=4	1+4=5	1+5=6	1+6=7	1+7=8	1+8=9	1+9=10
2+1=3	2+2=4	2+3=5	2+4=6	2+5=7	2+6=8	2+7=9	2+8=10	2+9=11
3+1=4	3+2=5	3+3=6	3+4=7	3+5=8	3+6=9	3+7=10	3+8=11	3+9=12
4+1=5	4+2=6	4+3=7	4+4=8	4+5=9	4+6=10	4+7=11	4+8=12	4+9=13
5+1=6	5+2=7	5+3=8	5+4=9	5+5=10	5+6=11	5+7=12	5+8=13	5+9=14
6+1=7	6+2=8	6+3=9	6+4=10	6+5=11	6+6=12	6+7=13	6+8=14	6+9=15
7+1=8	7+2=9	7+3=10	7+4=11	7+5=12	7+6=13	7+7=14	7+8=15	7+9=16
8+1=9	8+2=10	8+3=11	8+4=12	8+5=13	8+6=14	8+7=15	8+8=16	8+9=17
9+1=10	9+2=11	9+3=12	9+4=13	9+5=14	9+6=15	9+7=16	9+8=17	9+9=18

In[29]:=
```
(* 42.5 *)
Flatten[StringCases[#, ___ ~~ "i" ~~ ___ ~~ "e" ~~ ___] & /@ IntegerName /@ Range[0, 49]]
```

Out[29]=
```
{five, nine, thirteen, fifteen, sixteen, eighteen, nineteen,
 twenty-five, twenty-nine, thirty-one, thirty-three, thirty-five,
 thirty-seven, thirty-eight, thirty-nine, forty-five, forty-nine}
```

In[30]:=
```
(* 42.6 *) (* There is cause for dissatisfaction with this solution: *)
StringReplace[TextSentences[WikipediaData["computers"]][[1]],
 " " ~~ Shortest[x_] ~~ Shortest[y_] ~~ " " :>
  StringJoin[" ", ToUpperCase[x], ToUpperCase[y], " "]]
```

Out[30]=
```
A computer IS a machine that can BE programmed TO automatically
  carry out sequences OF arithmetic OR logical operations (computation).
```

In[31]:=
```
(* An alternative solution to 42.6 that also has cause for dissatisfaction: *)
StringRiffle[If[StringLength[#] == 2, ToUpperCase[#], #] & /@
  TextWords[TextSentences[WikipediaData["computers"]][[1]]], " "]
```

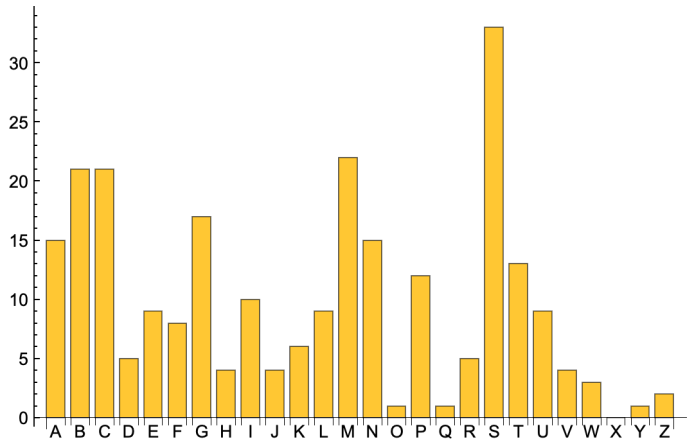
Out[31]=
```
A computer IS a machine that can BE programmed TO automatically
  carry out sequences OF arithmetic OR logical operations computation
```

In[33]:=
```
(* 42.7 *) countryNameCounts = Module[{allCountryNames =
    EntityValue[#, "Name"] & /@ EntityList[all countries, dependencies, and territories COUNTRIES]},
  Table[{ToUpperCase[FromLetterNumber[i]], Length[Select[allCountryNames,
      ToLowerCase[Characters[#][[1]]] == FromLetterNumber[i] &]]}, {i, 26}]]
```

Out[33]=
```
{{A, 15}, {B, 21}, {C, 21}, {D, 5}, {E, 9}, {F, 8}, {G, 17}, {H, 4},
 {I, 10}, {J, 4}, {K, 6}, {L, 9}, {M, 22}, {N, 15}, {O, 1}, {P, 12}, {Q, 1},
 {R, 5}, {S, 33}, {T, 13}, {U, 9}, {V, 4}, {W, 3}, {X, 0}, {Y, 1}, {Z, 2}}
```

In[34]:=
```
BarChart[countryNameCounts[[All, 2]], ChartLabels → countryNameCounts[[All, 1]]]
```

Out[34]=



In[35]:=
```
(* 42.8 *) (* Here is the thing we are supposed to simplify *)
Grid[Table[StringJoin[TextString[i], "^",
   TextString[j], "=", TextString[i^j]], {i, 5}, {j, 5}]]
```

Out[35]=
```
1^1=1  1^2=1   1^3=1   1^4=1    1^5=1
2^1=2  2^2=4   2^3=8   2^4=16   2^5=32
3^1=3  3^2=9   3^3=27  3^4=81   3^5=243
4^1=4 4^2=16  4^3=64  4^4=256 4^5=1024
5^1=5 5^2=25 5^3=125 5^4=625 5^5=3125
```

In[36]:=
```
(* Isn't this almost the same as 42.4? *)
Apply[StringTemplate["``^``=``"], Array[{#1, #2, #1^#2} &, {5, 5}], {2}] // Grid
```

Out[36]=
```
1^1=1  1^2=1   1^3=1   1^4=1    1^5=1
2^1=2  2^2=4   2^3=8   2^4=16   2^5=32
3^1=3  3^2=9   3^3=27  3^4=81   3^5=243
4^1=4 4^2=16  4^3=64  4^4=256 4^5=1024
5^1=5 5^2=25 5^3=125 5^4=625 5^5=3125
```