Looks great. 10/10. I responded to your comment on p. 18.

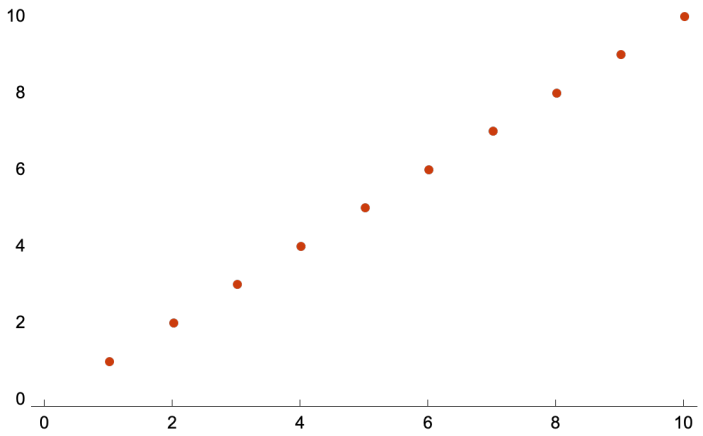
Rania —PS 8 (02.18.2025)

Sections 20, 21, 22 in EIWL3

Section 20: Options

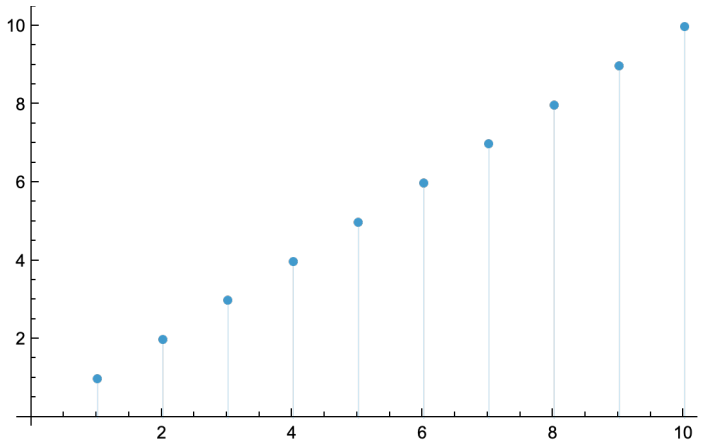
In[41]:=
```
(*20.1 A list plot of Range[10] themed for the web*)
ListPlot[Range[10], PlotTheme → "Web"]
```

Out[41]=



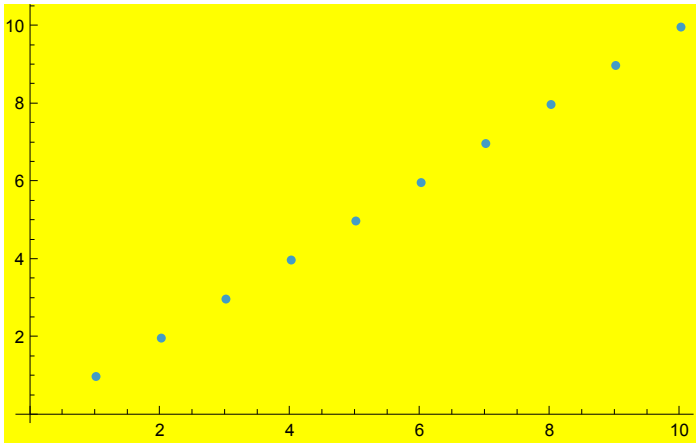
In[42]:=
```
(*20.2 A list plot of Range[10] with filling to the axis*)
ListPlot[Range[10], Filling → Axis]
```

Out[42]=



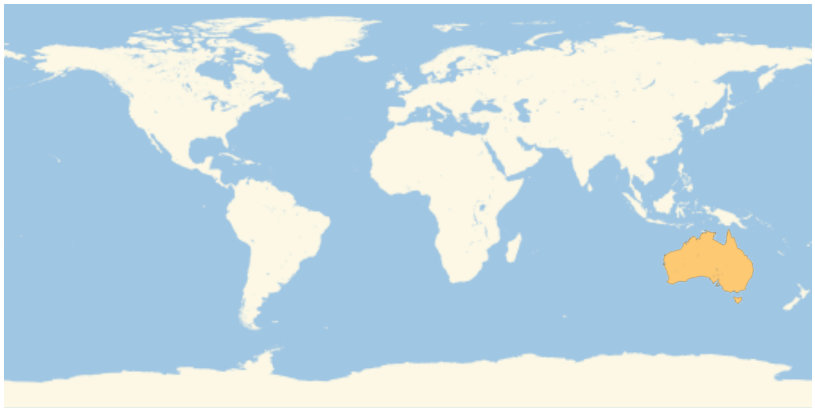
In[43]:=
```
(*20.3 A list plot of Range[10] with a yellow background*)
ListPlot[Range[10], Background → Yellow]
```

Out[43]=



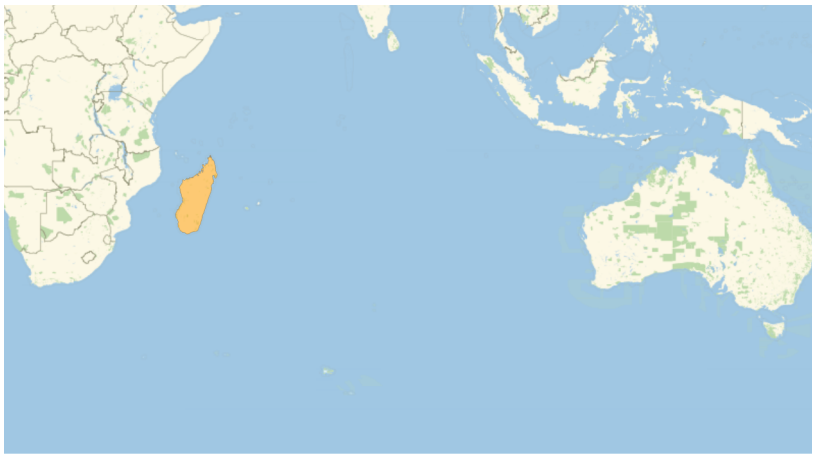
In[44]:=
```
(*20.4 Map of the world with Australia highlighted*)
GeoListPlot[ Australia COUNTRY , GeoRange → All]
```

Out[44]=



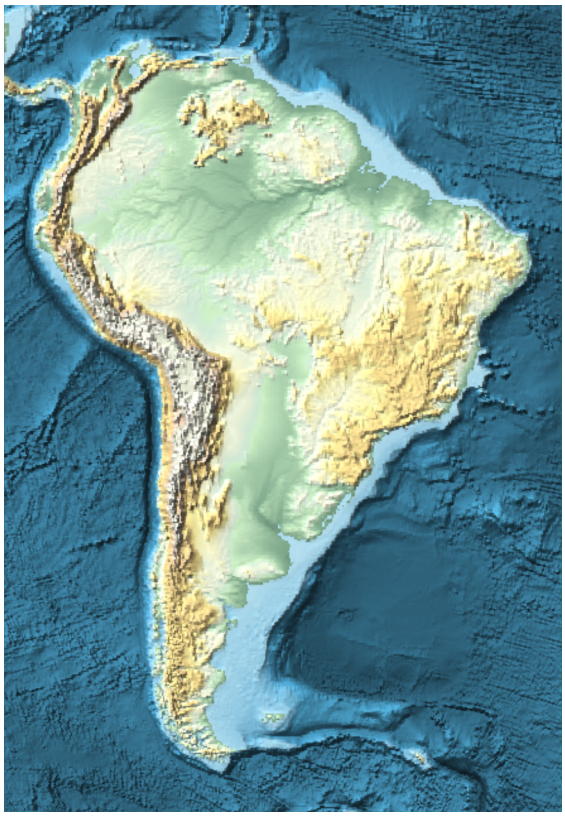
In[45]:=
```
(*20.5 Map of the Indian Ocean with Madagascar highlighted.*)
GeoListPlot[Madagascar COUNTRY, GeoRange → Indian Ocean OCEAN]
```

Out[45]=



In[46]:=
```
(*20.6 GeoGraphics to create a map of
 South America showing topography (relief map).*)
GeoGraphics[South America COUNTRIES, GeoBackground → "ReliefMap"]
```

Out[46]=



In[47]:=
```
(*20.7 Map of Europe with France, Finland, and Greece highlighted and labeled*)
GeoListPlot[{France COUNTRY, Finland COUNTRY, Greece COUNTRY},
 GeoRange → Europe GEOGRAPHIC REGION, GeoLabels → Automatic]
```

Out[47]=

In[48]:=
```
(*20.8 Positions of universities in the Ivy League, labeling each*)
(*during transfer season?!! really Wolfram....*)
GeoListPlot[The Ivy League UNIVERSITIES, GeoLabels → Automatic]
```

Out[48]=

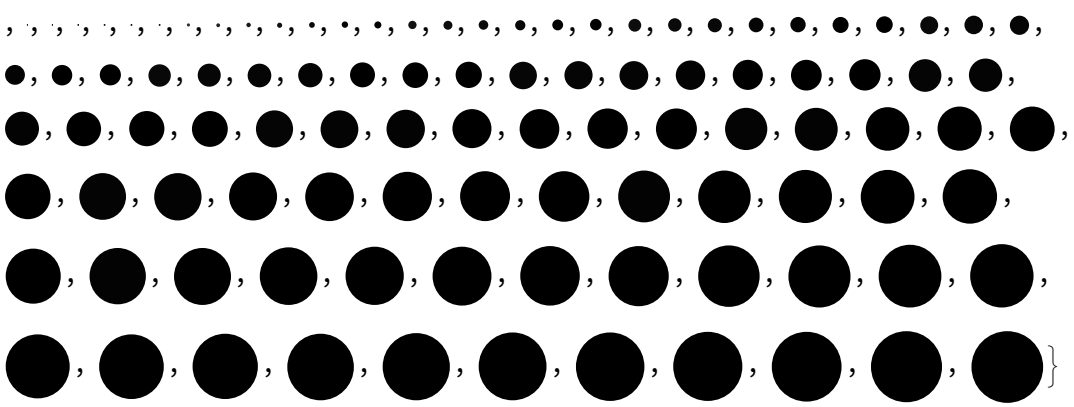
In[49]:=
```
(*20.9 12×12 multiplication table as
 a grid with white type on a black background*)
Grid[Table[i * j, {i, 12}, {j, 12}],
 Frame → All, ItemStyle → White, Background → Black]
```

Out[49]=

1	2	3	4	5	6	7	8	9	10	11	12
2	4	6	8	10	12	14	16	18	20	22	24
3	6	9	12	15	18	21	24	27	30	33	36
4	8	12	16	20	24	28	32	36	40	44	48
5	10	15	20	25	30	35	40	45	50	55	60
6	12	18	24	30	36	42	48	54	60	66	72
7	14	21	28	35	42	49	56	63	70	77	84
8	16	24	32	40	48	56	64	72	80	88	96
9	18	27	36	45	54	63	72	81	90	99	108
10	20	30	40	50	60	70	80	90	100	110	120
11	22	33	44	55	66	77	88	99	110	121	132
12	24	36	48	60	72	84	96	108	120	132	144

In[50]:=
```
(*20.10 List of 100 disks with random integer image sizes up to 40*)
Table[Graphics[Disk[], ImageSize → n], {n, 0, 40, .4}]
```

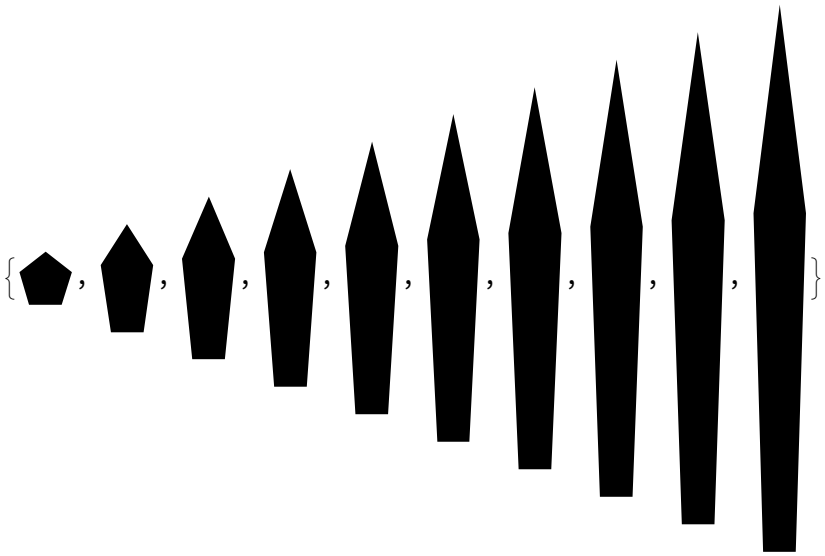
Out[50]=



Should have ImageSize->RandomInteger[{1,40}] not ImageSize->n.

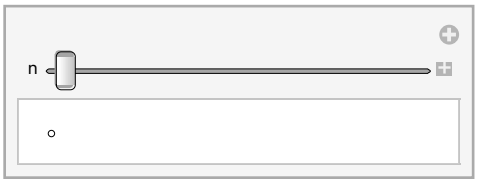
In[51]:=
```
(*20.11 A list of pictures of regular pentagons
 with image size 30 and aspect ratios from 1 to 10*)
Table[Graphics[RegularPolygon[5], ImageSize → 30, AspectRatio → n] , {n, 1, 10, 1}]
```

Out[51]=



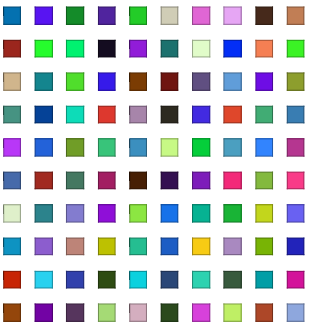
In[52]:=
```
(*20.12 Manipulate that varies the size of a circle between 5 to 500*)
Manipulate[Graphics[Circle[], ImageSize → n], {n, 5, 500, 1}]
```

Out[52]=



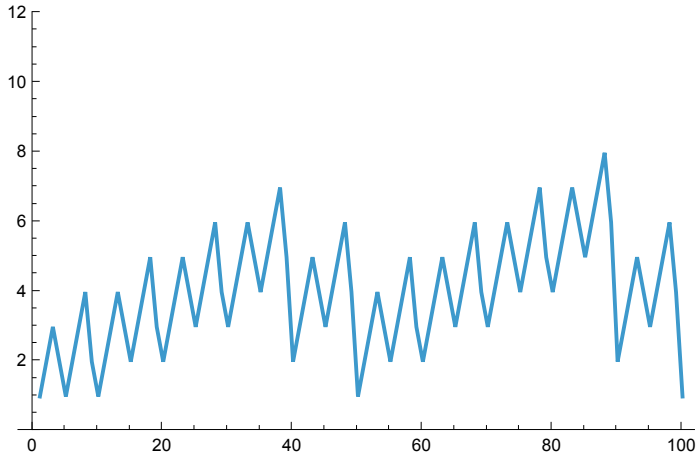
In[53]:=
```
(*20.13 A framed 10×10 grid of random colors*)
Grid[Table[RandomColor[], 10, 10]]
```

Out[53]=



In[54]:=
```
(*20.14 A line plot of the lengths of Roman numerals up to 100,
with a plot range that would be sufficient for all numerals up to 1000*)
ListLinePlot[Table[StringLength[RomanNumeral[n]], {n, 100}], PlotRange → {0, 12}]
```

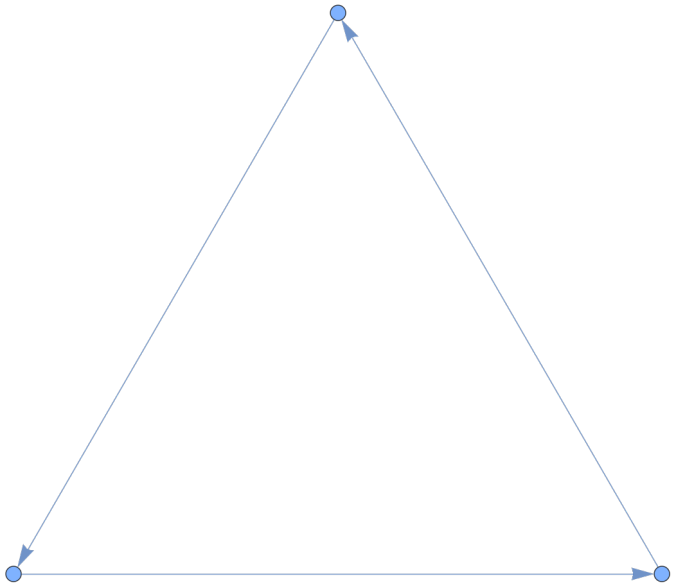
Out[54]=



Section 21: Graphs and Networks

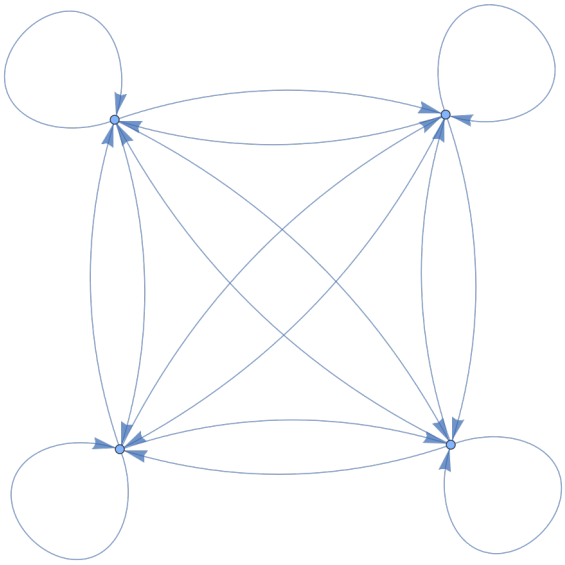
In[55]:=
```
(*21.1 Graph of a loop of 3 nodes*)
Graph[{1 → 2, 2 → 3, 3 → 1}]
```

Out[55]=



In[56]:=
```
(*21.2 Agraph with 4 nodes in which every node is connected*)
Graph[Flatten[Table[i → j, {i, 4}, {j, 4}]]]
```

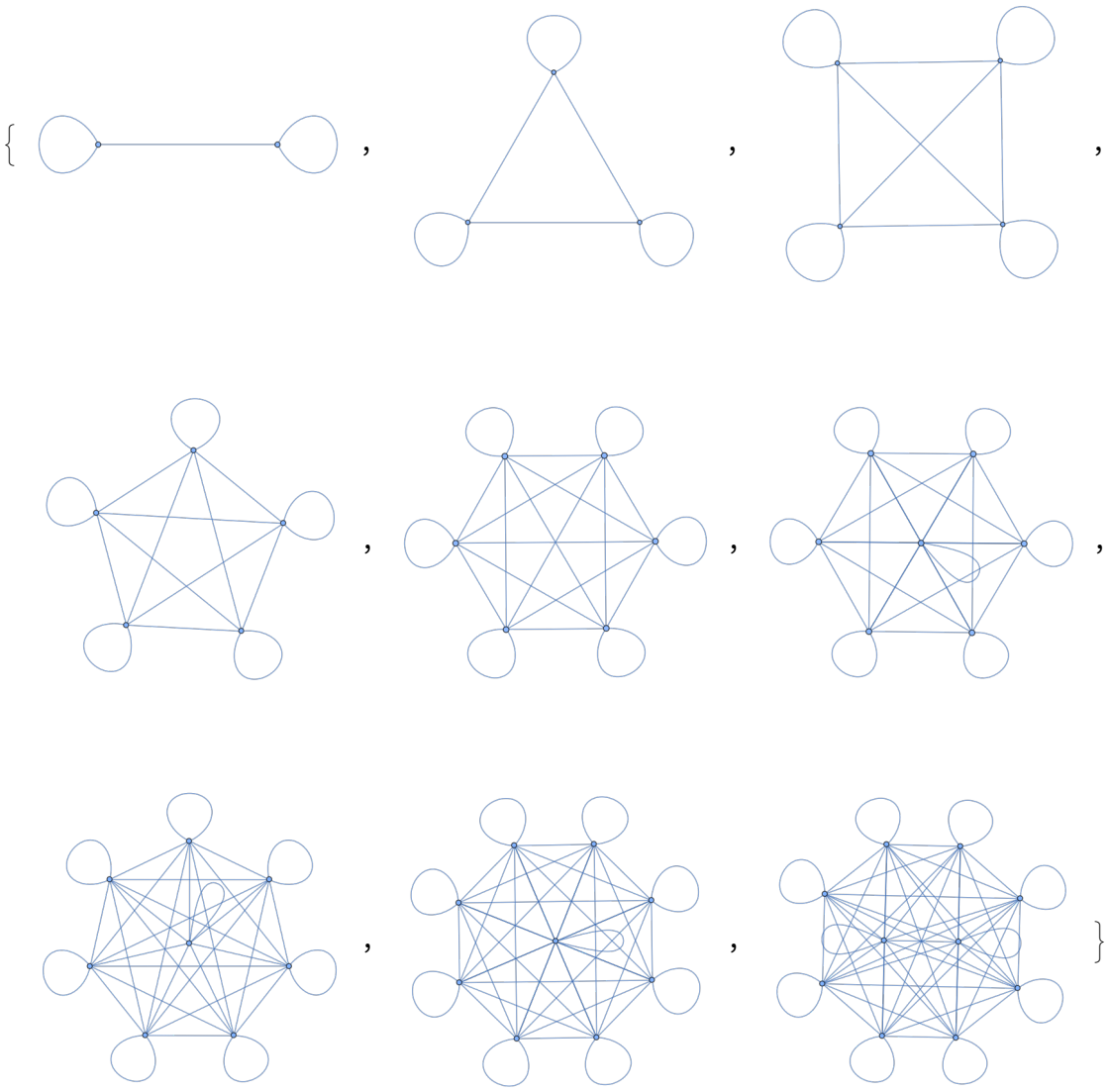
Out[56]=



I had a simpler interpretation of what was being asked on this one.

In[57]:=
```
(*21.3 Table of undirected graphs with between
 2 and 10 nodes in which every node is connected*)
Table[UndirectedGraph[Flatten[Table[i → j, {i, n}, {j, n}]]], {n, 2, 10, 1}]
```

Out[57]=



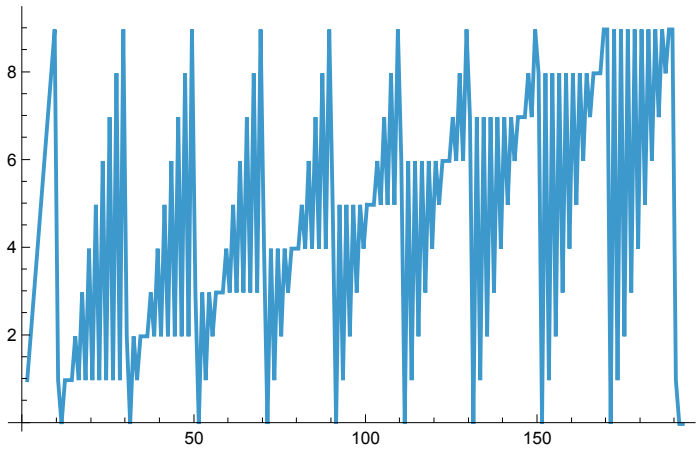
In[58]:=
```
(*21.4 Use Table and Flatten to get {1,2,1,2,1,2}*)
Flatten[Table[Range[2], 3]]
```

Out[58]=

```
{1, 2, 1, 2, 1, 2}
```

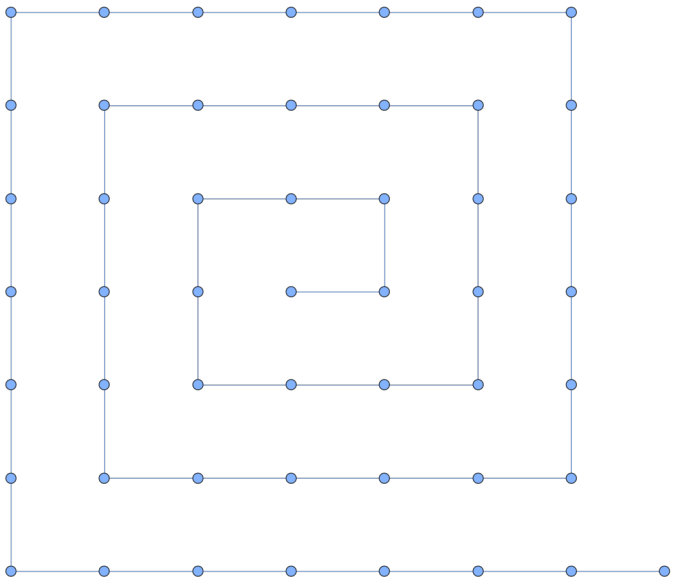
In[59]:=
```
(*21.5 A line plot of the result of concatenating all digits
 of all integers from 1 to 100 (i.e....,8,9,1,0,1,1,1,2,...)*)
ListLinePlot[Flatten[IntegerDigits[Range[100]]]]
```

Out[59]=



In[60]:=
```
(* 21.6 Make a graph with 50 nodes,in which node i connects to node i+1 *)
UndirectedGraph[Table[i → i + 1, {i, 49}]]
```

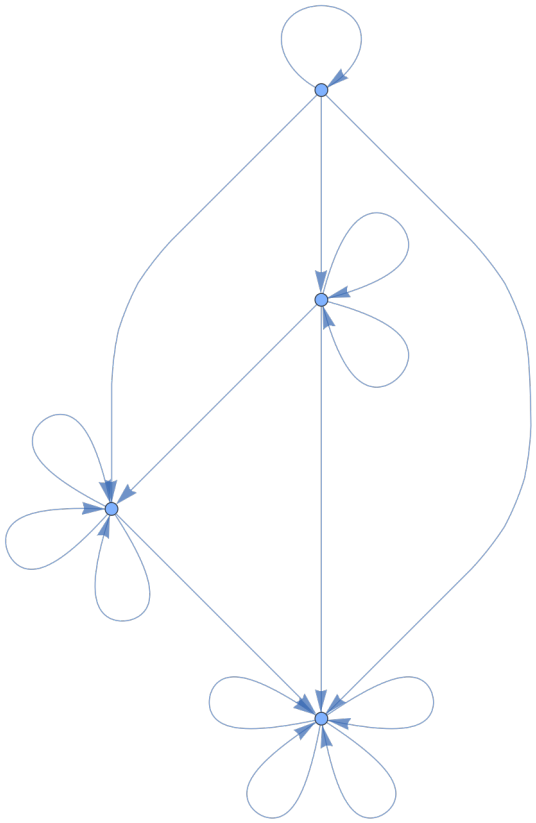
Out[60]=



Nice!

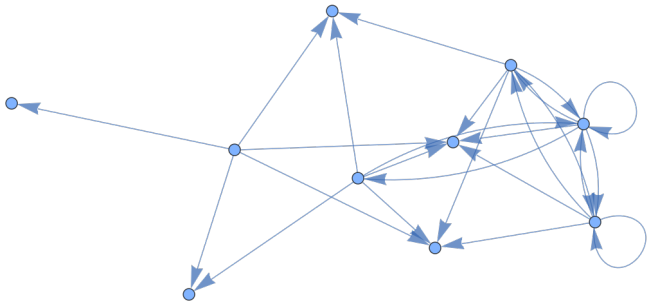
In[61]:=
```
(*21.7 Graph with 4 nodes,in which each connection connects i to Max[i,j]*)
Graph[Flatten[Table[i → Max[i, j], {i, 4}, {j, 4}]]]
```

Out[61]=



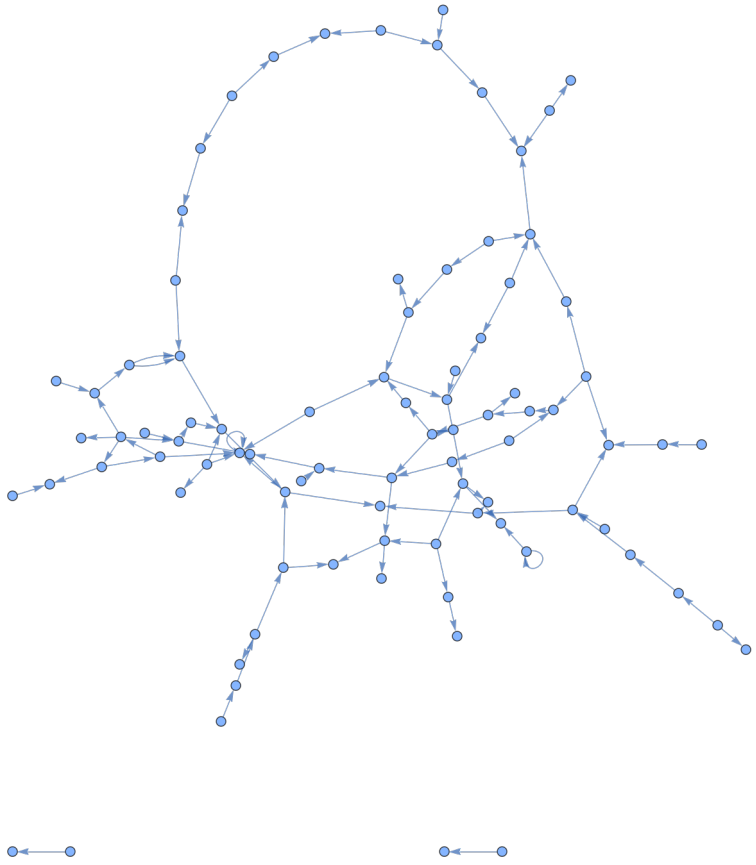
In[62]:=
```
(*21.8 Graph in which each connection connects i to j-i,
where i and j both range from 1 to 5.*)
Graph[Flatten[Table[i → j - i, {i, 5}, {j, 5}]]]
```

Out[62]=



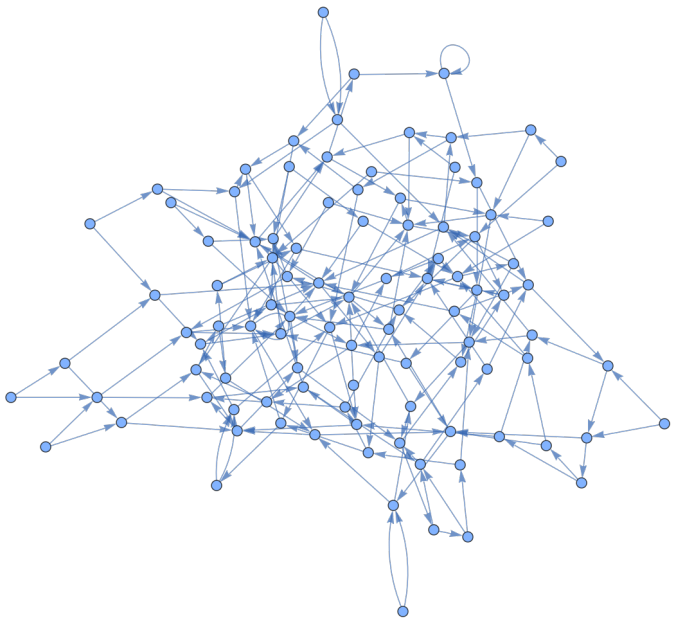
In[63]:=
```
(*21.9 Generate a graph with 100 nodes,
each with a connection going to one randomly chosen node.*)
Graph[Table[RandomInteger[99] → RandomInteger[99], 100]]
```

Out[63]=



In[64]:=
```
(*21.10 A graph with 100 nodes,each connecting to two randomly chosen nodes.*)
Graph[Flatten[Table[{n → RandomInteger[99], n → RandomInteger[99]}, {n, 0, 99}]]]
```

Out[64]=



In[65]:=
```
(*21.11 For the graph {1→2,2→3,3→4,4→1,3→1,2→2},
make a grid giving the shortest paths between every pair of nodes,
with the start node as row and end node as column.*)
Grid[Table[FindShortestPath[
   Graph[{1 → 2, 2 → 3, 3 → 4, 4 → 1, 3 → 1, 2 → 2}], n, j], {n, 4}, {j, 4}]]
```

Out[65]=

{1}	{1, 2}	{1, 2, 3}	{1, 2, 3, 4}
{2, 3, 1}	{2}	{2, 3}	{2, 3, 4}
{3, 1}	{3, 1, 2}	{3}	{3, 4}
{4, 1}	{4, 1, 2}	{4, 1, 2, 3}	{4}

Section 22: Machine Learning

In[66]:=
```
(*22.1 Identify what language the word “ajatella” comes from*)
LanguageIdentify["ajatella"]
```

Out[66]=

Finnish

In[67]:=
```
(*22.2 Apply ImageIdentify to an image of a tiger,
getting the image using ctrl+=*)
ImageIdentify[ tiger SPECIES SPECIFICATION [ image ]]
```

Out[67]=

tiger

In[68]:=
```
(*22.3 Make a table of image identifications for an image of a tiger,
blurred by an amount from 1 to 5*)
Table[ImageIdentify[Blur[tiger SPECIES SPECIFICATION[image], n]], {n, 1, 5}]
```

Out[68]=

{tiger, tiger, tiger, tiger, swift fox}

In[69]:=
```
(*22.4 Classify the sentiment of “I’m so happy to be here”.*)
Classify["Sentiment", "I'm so happy to be here"]
```

Out[69]=

Positive

In[70]:=
```
(*22.5 Find the 10 words in WordList[] that are nearest to “happy” *)
Nearest[WordList[], "happy", 10]
```

Out[70]=

{happy, haply, harpy, nappy, sappy, apply, campy, choppy, guppy, hairy}

In[71]:=
```
(*22.6 20 random numbers up to 1000 and find which 3 are nearest to 100.*)
Nearest[Table[RandomInteger[1000], 20], 100, 3]
```

Out[71]=

{64, 62, 44}

In[72]:=
```
(*22.7 A list of 10 random colors,and which 5 are closest to Red*)
Nearest[Table[RandomColor[], 10], Red, 5]
```

Out[72]=

{ , , , , }

In[73]:=
```
(*22.8 Of the first 100 squares,find the one nearest to 2000*)
Nearest[Table[x^2, {x, 1, 100}], 2000]
```

Out[73]=

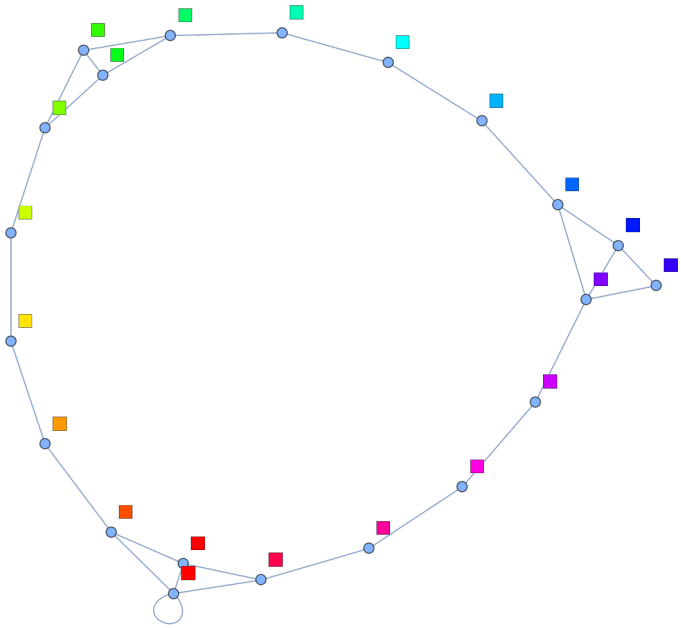
{2025}

In[74]:=
```
(*22.9 Find the 3 European flags nearest to the flag of Brazil.*)
Nearest[Europe COUNTRIES[flag], Brazil COUNTRY[flag], 3]
```

Out[74]=

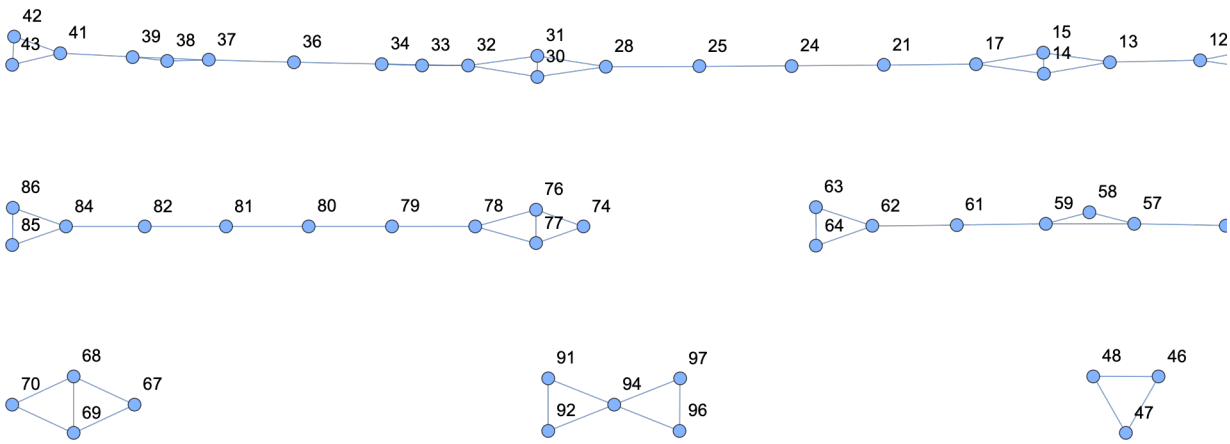
In[75]:=
```
(*22.10 A graph of the 2 nearest neighbors
 of each color in Table[Hue[h],{h,0,1,.05}]*)
colorList = Table[Hue[h], {h, 0, 1, .05}];
NearestNeighborGraph[colorList, 2, VertexLabels → All]
```

Out[76]=



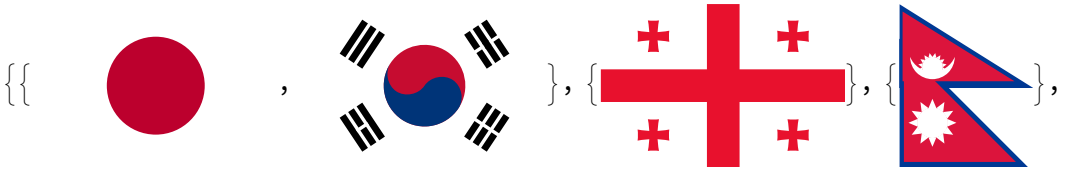
In[77]:=
```
(*22.11 A list of 100 random numbers from 0 to 100,
and make a graph of the 2 nearest neighbors of each one*)
randomNumto100 = Table[RandomInteger[100], 100];
NearestNeighborGraph[randomNumto100, 2, VertexLabels → All]
```

Out[78]=



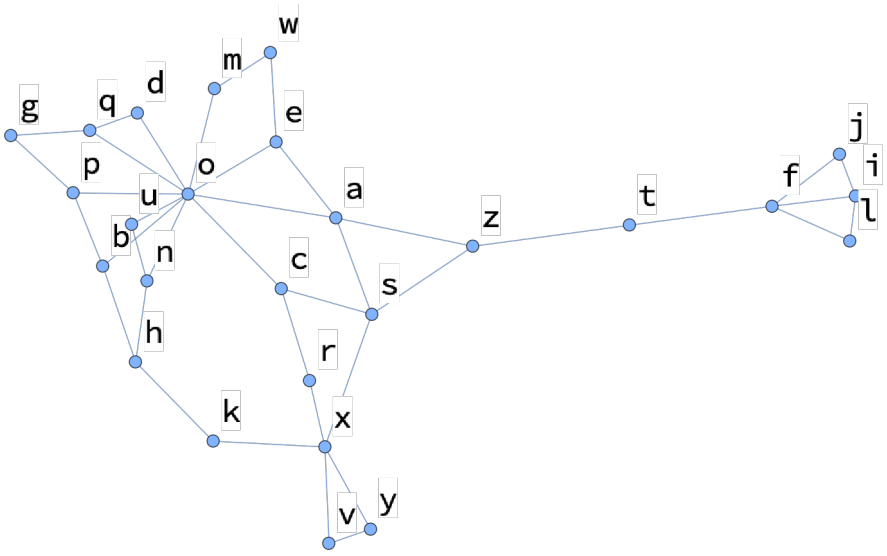
In[79]:=
```
(*22.12 Collect the flags of Asia into clusters of similar flags*)
FindClusters[ Asia COUNTRIES [ flag ] ]
```

Out[79]=



In[80]:=
```
(*22.13 Make raster images of the letters of the alphabet at size 20,
then make a graph of the 2 nearest neighbors of each one*)
NearestNeighborGraph[
 Table[Rasterize[FromLetterNumber[n], RasterSize → 20], {n, 26}],
 2, VertexLabels → Automatic]
```

Out[80]=



In[81]:=
```
(*22.14 Generate a table of the results from applying TextRecognize to the
 EdgeDetect of the word “programming” rasterized at sizes between 10 and 20*)
Table[TextRecognize[EdgeDetect[Rasterize["programming", RasterSize → n]]],
 {n, 10, 20, 1}]
Table[TextRecognize[EdgeDetect[Rasterize[Style["programming", n]]]], {n, 10, 20, 1}]

(*confusing..I don't understand the excercise nor can I get the desired output,
even when trying a Rasterize from 1 →
 100. Here are my two way of interperting the problem*)
```

Out[81]=
```
{, , , , , , , , , , }
```

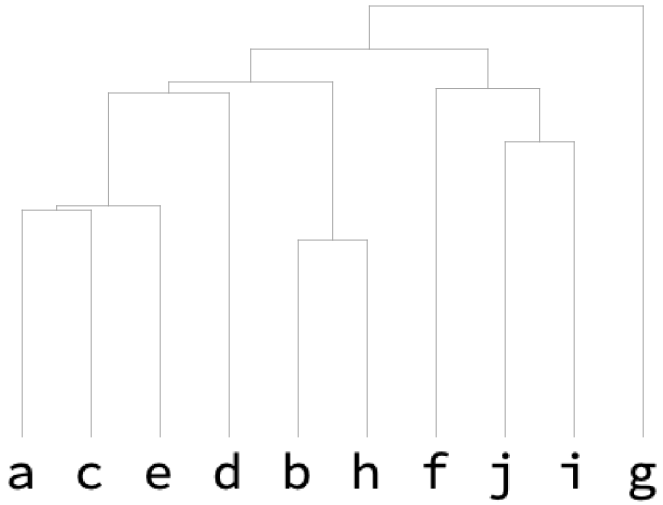
Out[82]=
```
{orogramming, programming, programming, programming, programming,
 programming, programming, programming, programming, programming, programming}
```

Here is my code. Your last result looks perfectly correct!

TextRecognize[EdgeDetect[Rasterize[Style["programming", size]]]], {size, 10, 20}]

In[83]:=
```
(*22.15 Make a dendrogram for images of the first 10 letters of the alphabet.*)
Dendrogram[Table[Rasterize[FromLetterNumber[n]], {n, 10}]]
```

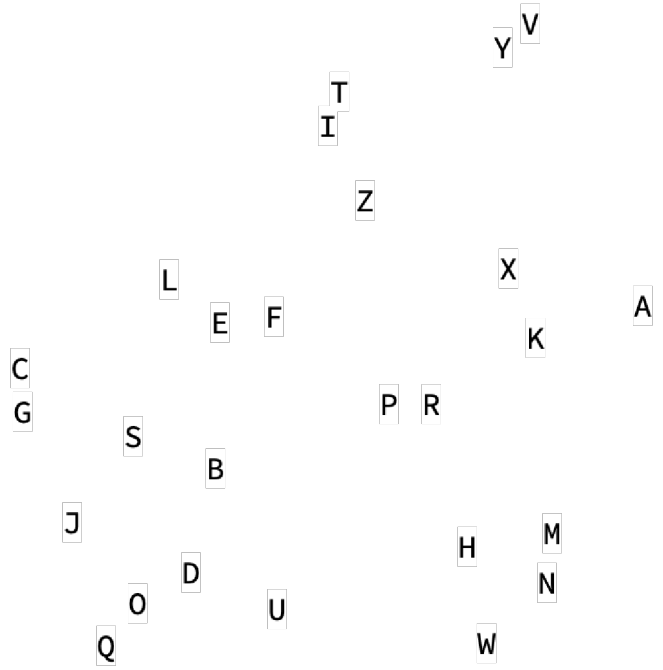
Out[83]=



Nice!

In[84]:=
```
(*22.16 Make a feature space plot for the uppercase letters of the alphabet.*)
FeatureSpacePlot[Table[Rasterize[ToUpperCase[FromLetterNumber[n]]], {n, 26}]]
```

Out[84]=



Also nice!

In[85]:=