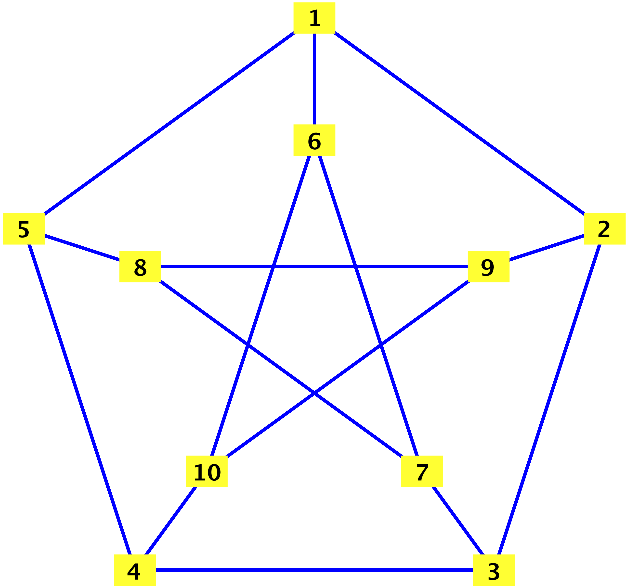
```
with(GraphTheory) :
G := Graph(5, {{1, 2}, {2, 3}, {3, 4}, {4, 1}, {3, 5}, {4, 5}});
            Graph 1: an undirected unweighted graph with 5 vertices and 6 edge(s)            (1)
H := Graph(5, {[1, 2], [2, 3], [3, 4], [4, 3], [4, 1], [3, 5], [4, 5]});
              Graph 2: a directed unweighted graph with 5 vertices and 7 arc(s)              (2)
IsEulerian(G);
                                         false                                               (3)
IsPlanar(G,'Faces');
                                         true                                                (4)
Faces;
                         [[4, 1, 2, 3], [3, 2, 1, 4, 5], [4, 3, 5]]                          (5)
P := SpecialGraphs[PetersenGraph]( );
           Graph 3: an undirected unweighted graph with 10 vertices and 15 edge(s)           (6)
DrawGraph(P);
```

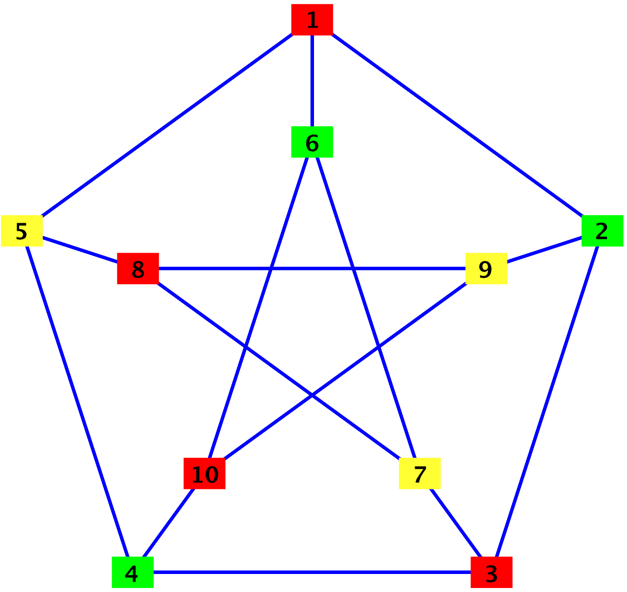


```
IsPlanar(P);
                                         false                                               (7)
IsVertexColorable(P, 3,'C')
                                         true                                                (8)
```

```
C;
```

[[1, 3, 8, 10], [2, 4, 6], [5, 7, 9]] **(9)**

```
HighlightVertex(P, [1, 3, 8, 10], red);
HighlightVertex(P, [2, 4, 6], green);
DrawGraph(P)
```



```
CP := ChromaticPolynomial(P, lambda);
```

$$\lambda\,(\lambda-1)\,(\lambda-2)\,\left(\lambda^{7}-12\,\lambda^{6}+67\,\lambda^{5}-230\,\lambda^{4}+529\,\lambda^{3}-814\,\lambda^{2}+775\,\lambda-352\right) \qquad \textbf{(10)}$$

```
eval(CP, lambda = 3);
```

120 **(11)**

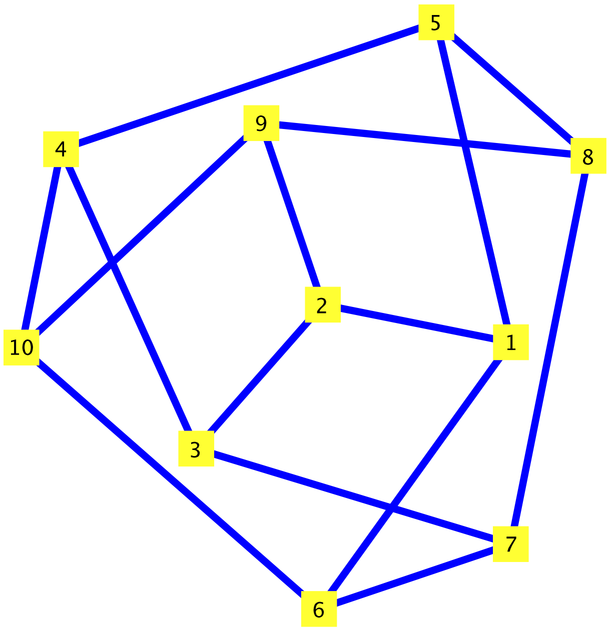
```
eval(CP, lambda = 2);
```

0 **(12)**

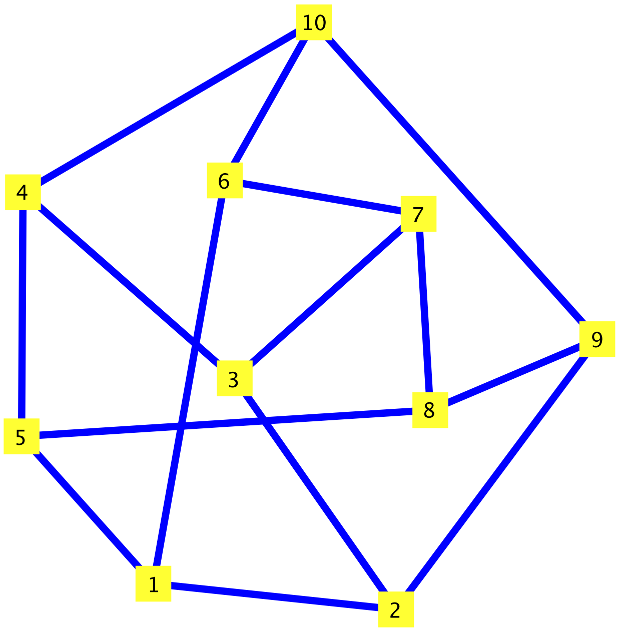
```
with(SpecialGraphs) :
P := PetersenGraph( );
```

Graph 4: an undirected unweighted graph with 10 vertices and 15 edge(s) **(13)**

```
DrawGraph(P, style = spring);
```



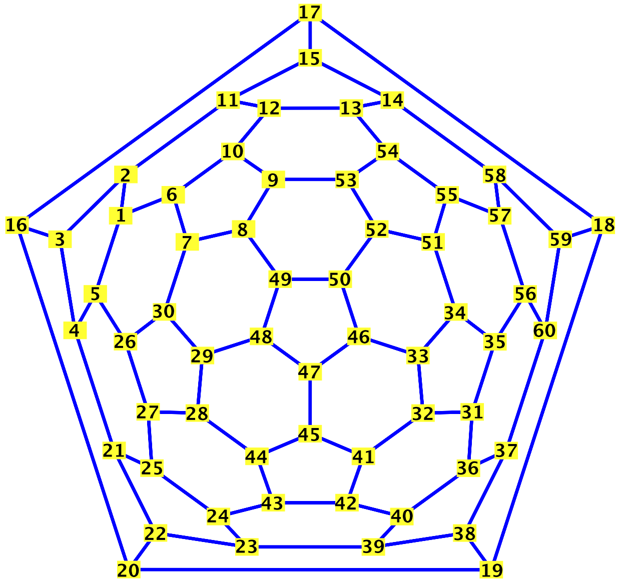
DrawGraph(*P*, *style* = *spring*, *redraw*);



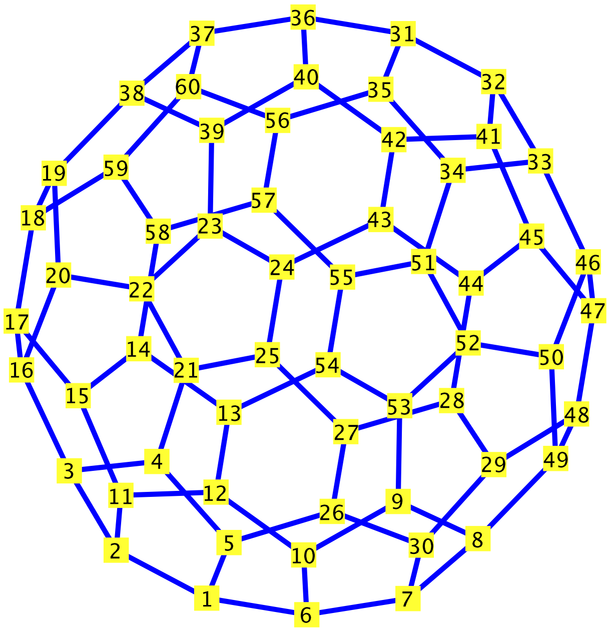
```
S := SoccerBallGraph( );
```

Graph 5: an undirected unweighted graph with 60 vertices and 90 edge(s) **(14)**

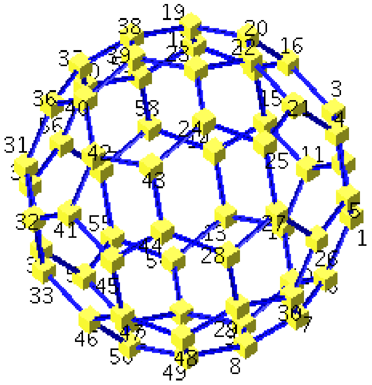
```
DrawGraph(S);
```



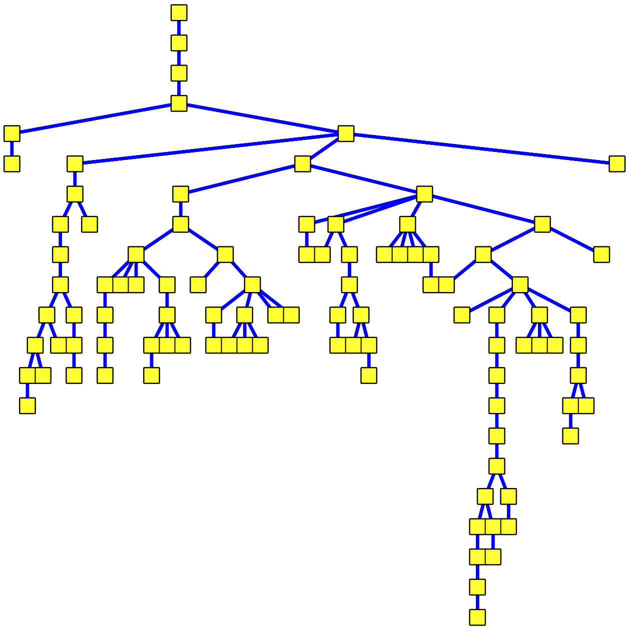
```
DrawGraph(S, style=spring);
```



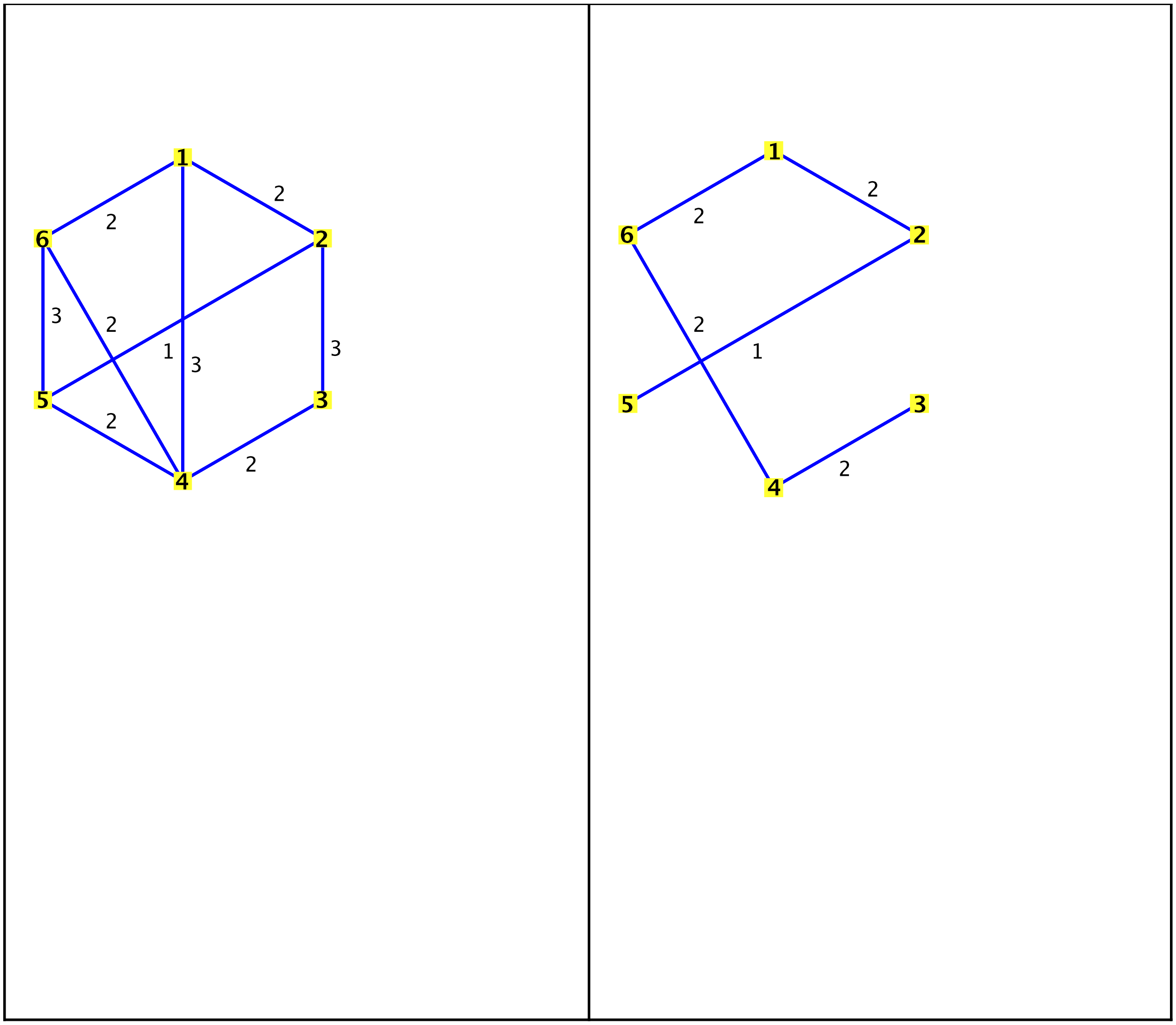
```
DrawGraph(S, style = spring, dimension = 3);
```



```
with(RandomGraphs) :
T := RandomTree(100) :
DrawGraph(T)
```

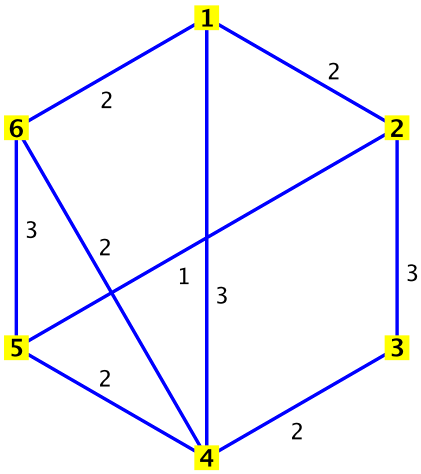


```
#KRUSKAL'S ALGORITHM
with(GraphTheory) :
G := Graph(6, {[{1, 2}, 2], [{2, 3}, 3], [{3, 4}, 2], [{1, 4}, 3], [{4, 5}, 2], [{5,
6}, 3], [{4, 6}, 2], [{2, 5}, 1], [{1, 6}, 2]});
T := MinimalSpanningTree(G);
                  Graph 6: an undirected weighted graph with 6 vertices and 9 edge(s)
                  Graph 7: an undirected weighted graph with 6 vertices and 5 edge(s)          (15)
DrawGraph([G, T], style=circle);
```



```
KruskalsAlgorithm(G, animate);
```

Input Graph



At each iteration Kruskal's Algorithm adds the cheapest edge that doesn't create a cycle in the tree.