Display and define the "Edelman" matrix which has entries ±1.

The determinant in exact arithmetic is even as it is a sum of an even number, factorial(27), terms all of which are ±1.

In [251]:

```
# For compactness create a 27² =  729  string of plusses and minuses
EdelmanData = "++---+-+---+-+--+++-+---++++-+-++-+++-++++-++--+-+-+--+--+++-+-------+-----+-++++++---+--++---++-++-+
-++++-+--+--+-+++++-+-+++++--++++--++--+-++-+---+-------+------+---+-----++++++++-++-++++-++-+-------++--+----+--+++
-+-+-+++--+-----+-+---++-+-------+-++----++-----+--+-+-+--++-+++---+-++-+++-+--+-+--++--++--+++---+++-+-+++--+++-+++
+---+---+++++--++-+-+-++-++++--+--+--+--++--+--++--+-----++---+-+-----+++-+---++++-++----++-+++-+-+-----++-+-+----++
++--++-+-+++-+-++++++++++-++----+-+++--++--++-++---++----++++-------+--++-++-++-+-----+-+-+----+++++--+++++--+--++--
+---++++-+-++-+-++++-+++++-++-++--++-++++-+--+++++-++++-+---++--++++--++-+++-+-++-+-++-+--+++-++-+-+-+----+-++-++-++
++-----+--+++--+-++---+--------+----+-+--+-+-+-+"

# print row by row for human viewing
for i=0:26
    println(EdelmanData[27*i+(1:27) ])
end

# Create integer Matrix
Edelman = ones(Int,27,27)
for i=1:27, j=1:27
   if EdelmanData[27*(i-1)+j].=='-'
        Edelman[i,j] = -1
    end
end
Edelman
```

```
++---+-+---+-+--+++-+---+++
+-+-++-+++-++++-++--+-+-+--
+--+++-+-------+-----+-++++
++---+--++---++-++-+-++++-+
--+--+-+++++-+-+++++--++++-
-++--+-++-+---+-------+----
--+---+-----++++++++-++-+++
+-++-+-------++--+----+--++
+-+-+-+++--+-----+-+---++-+
-------+-++----++-----+--+-
+-+--++-+++---+-++-+++-+--+
-+--++--++--+++---+++-+-+++
--+++-++++---+---+++++--++-
+-+-++-++++--+--+--+--++--+
--++--+-----++---+-+-----++
+-+---++++-++----++-+++-+-+
-----++-+-+----++++--++-+-+
++-+-++++++++++-++----+-+++
--++--++-++---++----++++---
----+--++-++-++-+-----+-+-+
----+++++--+++++--+--++--+-
--++++-+-++-+-++++-+++++-++
-++--++-++++-+--+++++-++++-
+---++--++++--++-+++-+-++-+
-++-+--+++-++-+-+-+----+-++
-++-++++-----+--+++--+-++--
-+--------+----+-+--+-+-+-+
```

Out[251]:

```
27×27 Array{Int64,2}:
  1   1  -1  -1  -1   1  -1   1  -1  …   1  -1   1  -1  -1  -1   1   1   1
  1  -1   1  -1   1   1  -1   1   1     -1  -1   1  -1   1  -1   1  -1  -1
  1  -1  -1   1   1   1  -1   1  -1     -1  -1  -1   1  -1   1   1   1   1
  1   1  -1  -1  -1   1  -1  -1   1     -1   1  -1   1   1   1   1  -1   1
 -1  -1   1  -1  -1   1  -1   1   1      1   1  -1  -1   1   1   1   1  -1
 -1   1   1  -1  -1   1  -1   1   1  …  -1  -1  -1  -1   1  -1  -1  -1  -1
 -1  -1   1  -1  -1  -1   1  -1  -1      1   1  -1   1   1  -1   1   1   1
  1  -1   1   1  -1   1  -1  -1  -1     -1  -1  -1  -1   1  -1  -1   1   1
  1  -1   1  -1   1  -1   1   1   1     -1   1  -1  -1  -1   1   1  -1   1
 -1  -1  -1  -1  -1  -1  -1   1  -1     -1  -1  -1  -1   1  -1  -1   1  -1
  1  -1   1  -1  -1   1   1  -1   1  …  -1   1   1   1  -1   1  -1  -1   1
 -1   1  -1  -1   1   1  -1  -1   1      1   1   1  -1   1  -1   1   1   1
 -1  -1   1   1   1  -1   1   1   1      1   1   1   1  -1  -1   1   1  -1
  ⋮                   ⋮                ⋱              ⋮                  ⋮
  1  -1   1  -1  -1  -1   1   1   1  …   1  -1   1   1   1  -1   1  -1   1
 -1  -1  -1  -1  -1   1   1  -1   1      1  -1  -1   1   1  -1   1  -1   1
  1   1  -1   1  -1   1   1   1   1     -1  -1  -1  -1   1  -1   1   1   1
 -1  -1   1   1  -1  -1   1   1  -1     -1  -1   1   1   1   1  -1  -1  -1
 -1  -1  -1  -1   1  -1  -1   1   1     -1  -1  -1  -1   1  -1   1  -1   1
 -1  -1  -1  -1   1   1   1   1   1  …   1  -1  -1   1   1  -1  -1   1  -1
 -1  -1   1   1   1   1  -1   1  -1     -1   1   1   1   1   1  -1   1   1
 -1   1   1  -1  -1   1   1  -1   1      1   1   1  -1   1   1   1   1  -1
  1  -1  -1  -1   1   1  -1  -1   1      1   1  -1   1  -1   1   1  -1   1
 -1   1   1  -1   1  -1  -1   1   1      1  -1  -1  -1  -1   1  -1   1   1
 -1   1   1  -1   1   1   1   1  -1  …   1  -1  -1   1  -1   1   1  -1  -1
 -1   1  -1  -1  -1  -1  -1  -1  -1     -1  -1   1  -1   1  -1   1  -1   1
```

In [272]:
```
# Use Julia's built in det which computes LU and takes ±product of the pivots
@printf("%0.2f",det(Edelman))
```

```
839466457497597.75
```

In [321]:
```
# Use BigInt's -- pushes the rounding to the very end
@printf("%0.90f",det(BigInt.(Edelman)))
```

```
839466457497600.000000000000000000000000000000000000000000000000000000000000583407682299482035044756000504
```

In [284]:
```
# Use Python's symbolic package from Julia
# Pkg.add("SymPy")
using SymPy
det(Sym.(Edelman))
```

Out[284]: 839466457497600

In [293]:
```
# use svd
@printf("%0.2f",prod(svdvals(Edelman)))
```

```
839466457497602.50
```

In [319]:
```
# use svd in backwards order
@printf("%0.2f",prod(svdvals(Edelman)[end:-1:1]))
```

```
839466457497602.75
```

In [311]:
```
# use pivots from lufact
@printf("%0.2f",-prod(diag(lufact(Edelman)[:U])))
```

```
839466457497597.13
```

In [318]:
```
# use backward pivots from lufact
@printf("%0.2f",-prod(diag(lu(Edelman)[2])[end:-1:1]))
```

```
839466457497597.50
```

In [336]:
```
s = svdvals(Edelman)
trials = 10000000
dets = [prod(s[randperm(27)])-839466457497600 for i=1:trials]
minimum(dets),mean(dets), maximum(dets)
```

Out[336]:
```
(1.25, 2.3218399375, 3.375)
```

In [337]:
```
using Plots
gr()
```

Out[337]:
```
Plots.GRBackend()
```

In [339]:
```
histogram(dets,label="count")
```

Out[339]: