```
(* MATH7502 Prac 2. *)

(* Construct a 3 by 3 matrix A. Take the trace, determinant,
and inverse. Use indexing to calculate the determinant in other ways. *)

A = {{1, 2, 3}, {4, 1, 6}, {7, 8, 1}};
MatrixForm[A]
(* In Julia, A = [1 2 3; 4 1 6; 7 8 1]              *)

 (* Trace of A: *)
Tr[A]
(* In Julia, tr(A)  *)

(* Determinant of A: *)
Det[A]
(* In Julia, det(A)  *)

(* Take the determinant of A using minors and indexing. *)
{A[[1, 1]], A[[1, 2]], A[[1, 3]]}
p = A[[2 ;; 3, 2 ;; 3]];
q = Transpose[ArrayFlatten[{A[[2 ;; 3]][[All, 1]], A[[2 ;; 3]][[All, 3]]}]];
r = A[[2 ;; 3, 1 ;; 2]];
{MatrixForm[p], MatrixForm[q], MatrixForm[r]}
A[[1, 1]] Det[p] - A[[1, 2]] Det[q] + A[[1, 3]] Det[r]

(* In Julia,
A[1,1]*det(A[2:3,2:3])-A[1,2]*det(A[2:3,[1,3]])+A[1,3]*det(A[2:3,1:2])     *)
```

Out[606]//MatrixForm=

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 1 & 6 \\ 7 & 8 & 1 \end{pmatrix}$$

Out[607]= 3

Out[608]= 104

Out[609]= {1, 2, 3}

Out[613]= $\left\{\begin{pmatrix} 1 & 6 \\ 8 & 1 \end{pmatrix}, \begin{pmatrix} 4 & 6 \\ 7 & 1 \end{pmatrix}, \begin{pmatrix} 4 & 1 \\ 7 & 8 \end{pmatrix}\right\}$

Out[614]= 104

In[621]:=
```
(* Inverse *)
Inverse[A] // MatrixForm
N[Inverse[A]] // MatrixForm

(* In Julia,
Ai = inv(A)
  inv(Ai) *)
```

Out[621]//MatrixForm=

$$\begin{pmatrix} -\frac{47}{104} & \frac{11}{52} & \frac{9}{104} \\ \frac{19}{52} & -\frac{5}{26} & \frac{3}{52} \\ \frac{25}{104} & \frac{3}{52} & -\frac{7}{104} \end{pmatrix}$$

Out[622]//MatrixForm=

$$\begin{pmatrix} -0.451923 & 0.211538 & 0.0865385 \\ 0.365385 & -0.192308 & 0.0576923 \\ 0.240385 & 0.0576923 & -0.0673077 \end{pmatrix}$$

In[662]:=
```
(* Consider the matrix equation C x+(x^T D)^T = x+b, where
  C = (1 2; 3 4), D = (-1 0; 2 1), b = (1; 2). Find x. *)

(* Simplifying,  C x + D^T x - I x = b, and
  (C + D^T - I) x = b. Now we can solve for x.    *)

A = {{1, 2}, {3, 4}} + Transpose[{{-1, 0}, {2, 1}}] - {{1, 0}, {0, 1}};
b = {{1}, {2}};
x = LinearSolve[A, b];
N[x] // MatrixForm

(* In Julia,
C=[1 2;
    3 4]
   D=[-1 0;
     2 1]
    b=[1;
    2];
A = C + D' - I
    xSol=A\ b
*)
```

$$C\,x+\left(x^T D\right)^T = x+b,\quad C = \begin{pmatrix}1 & 2\\ 3 & 4\end{pmatrix},\ D = \begin{pmatrix}-1 & 0\\ 2 & 1\end{pmatrix},\ b = \begin{pmatrix}1\\ 2\end{pmatrix}$$

Out[665]//MatrixForm=

$$\begin{pmatrix} 0.25 \\ 0.3125 \end{pmatrix}$$

```
(* Build a 5 by 1 matrix u of 1s. Build a 5 by 1 matrix v
 consisting of integers 1 to 5. Take their dot product in two ways. *)

u = ConstantArray[1, {5, 1}];
v = Table[{j}, {j, 1, 5}];
{Flatten[u].Flatten[v], (Transpose[u].v)[[1, 1]]}

(* In Julia,

u=fill(1,5)
v=1:5;
dot(u,v), u'*v              *)
```

Out[689]= {15, 15}

In[706]:=
```
(* List the integers from 1 to 21,
and form a 3 by 7 matrix in which we count from top to bottom. *)

A = Transpose[Partition[Table[j, {j, 1, 21}], 3]];
MatrixForm[A]

(* In Julia,
A=reshape(collect(1:21),3,7)  *)
```

Out[707]//MatrixForm=

$$\begin{pmatrix} 1 & 4 & 7 & 10 & 13 & 16 & 19 \\ 2 & 5 & 8 & 11 & 14 & 17 & 20 \\ 3 & 6 & 9 & 12 & 15 & 18 & 21 \end{pmatrix}$$

In[768]:=
```
(* Take the rank of the matrix A. *)
MatrixRank[A]

(* In Julia,
rank(A)                                                      *)

(* Take the row reduced echelon form of A (Gauss-Jordan) *)
A = RowReduce[A];
MatrixForm[A]

(* In Julia,
import Pkg;Pkg.add("RowEchelon")
 using RowEchelon
 rref(A)                                                     *)

(* Columns 3 to 7 are linear combinations of the first two columns. The dimension
 of the row space is equal to the dimension of the column space, the rank. *)

(* For example, Column 3 is -1*Column 1 + 2*Column 2. *)
B = A[[All, 1 ;; 2]];
X = Partition[A[[All, 3]], 1];
MatrixForm[B]
MatrixForm[X]
LinearSolve[B, X] // MatrixForm

(* In Julia, *)
(*  x3=A[:,1:2]\ A[:,3]   *)
```

Out[768]= 2

Out[770]//MatrixForm=

$$\begin{pmatrix} 1 & 0 & -1 & -2 & -3 & -4 & -5 \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Out[773]//MatrixForm=

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix}$$

Out[774]//MatrixForm=

$$\begin{pmatrix} -1 \\ 2 \\ 0 \end{pmatrix}$$

Out[775]//MatrixForm=

$$\begin{pmatrix} -1 \\ 2 \end{pmatrix}$$

In[794]:=
```
(* Construct a 3 by 1 matrix v with entries -1,
take various norms of v, then contruct a corresponding unit vector. *)
v = ConstantArray[-1, {3, 1}];
(* Euclidean norm*)
Norm[v]
(* Sum of absolute values *)
Norm[v, 1]
(* Max of absolute values *)
Norm[v, ∞]
Normalize[Flatten[v]]

(* In Julia,
v=-ones(3)
   norm(v,1)
   norm(v,Inf)
   norm([1,2,-8,3],Inf)
   v=normalize(ones(3))    *)
```

Out[795]= $\sqrt{3}$

Out[796]= 3

Out[797]= 1

Out[798]= $\left\{-\frac{1}{\sqrt{3}}, -\frac{1}{\sqrt{3}}, -\frac{1}{\sqrt{3}}\right\}$

```
(* Norms of random matrices. *)
(* In Julia,

import Pkg; Pkg.add("Distributions")

 using Plots,Distributions
N=300
data=zeros(2,N)
normData=zeros(2,N)

for i in 1:N
    x=rand(Uniform(-3,3),2)
    data[:,i]=x
    normData[:,i]=x/norm(x)
end

         scatter(data[1,:],data[2,:],legend=false,c= :black)
for i in 1:N
plot!([0,data[1,i]],[0,data[2,i]],c= :green)
end
scatter!(normData[1,:],normData[2,:],c= :red,size=(400,400))                *)
```

```
(* The Lagrange polynomial:
   L(X) =
```
$\sum_{j=0}^{n} y_j \left(\left(\prod_{i=0}^{j-1} \frac{X-x_i}{x_j-x_i}\right)\left(\prod_{i=j+1}^{n} \frac{X-x_i}{x_j-x_i}\right)\right)$
```
 is the same as the Vandermonde matrix interpolation. This
    is a symbolic calculation verifying that for n = 3 but we get a hint that
   this holds generally since the determinant of this matrix is a discriminant,
which also appears in the denominator of the interpolation formula.  *)
```

$$S = \text{Solve}\left[\begin{pmatrix} 1 & x_0 & x_0^2 & x_0^3 \\ 1 & x_1 & x_1^2 & x_1^3 \\ 1 & x_2 & x_2^2 & x_2^3 \\ 1 & x_3 & x_3^2 & x_3^3 \end{pmatrix} . \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{pmatrix} == \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{pmatrix}, \{c_0, c_1, c_2, c_3\}\right];$$

```
V = S[[1, 1, 2]] + S[[1, 2, 2]] X + S[[1, 3, 2]] X^2 + S[[1, 4, 2]] X^3;
n = 3;
```

$$\text{lg} = \sum_{j=0}^{n} y_j \left(\left(\prod_{i=0}^{j-1} \frac{X - x_i}{x_j - x_i}\right)\left(\prod_{i=j+1}^{n} \frac{X - x_i}{x_j - x_i}\right)\right);$$

```
L = Collect[Expand[lg], X];
Factor[Expand[L - V]]
```

Out[891]= 0

In[998]:=
```
(* From From[SWJ] Chapter 8, pp. 301 : *)
xVals = {-2, 3, 5, 6, 12, 14};
yVals = {7, 2, 9, 3, 12, 3};
n = Length[xVals]
V = Table[xVals[[i + 1]]^j, {i, 0, n - 1}, {j, 0, n - 1}];
MatrixForm[V]
(* The coefficients c0 to c5 *)
S = Flatten[N[LinearSolve[V, Partition[yVals, 1]]]]
(* The Lagrange interpolation polynomial. *)
f1 = Sum[S[[j + 1]] x^j, {j, 0, n - 1}]

(* obtained via least squares... *)
{beta0, beta1} = {5.525, 0.075};
f2 = beta0 + beta1 * x;

(* The data as points. *)
points = Table[{xVals[[j]], yVals[[j]]}, {j, 1, n}];

(* Plots and combined plots. *)
G = Plot[{f1, f2}, {x, First[xVals], Last[xVals]}, PlotRange → All];
H = ListPlot[points, PlotStyle → PointSize[0.02]];
Show[G, H]


(* In Julia,

import Pkg;Pkg.add("PyPlot")

 using Plots; pyplot()

  xVals=[-2,3,5,6,12,14]
yVals=[7,2,9,3,12,3]
n=length(xVals)

     V=[xVals[i+1]^(j) for i in 0:n-1,j in 0:n-1] #Vandermonde matrix-explain
c=V\ yVals
xGrid=-5:0.01:20
f1(x)=c'*[x^i for i in 0:n-1]

        beta0, beta1=5.525, 0.075 #obtained via least squares...
f2(x) = beta0 + beta1*x

   plot(xGrid,f1.(xGrid),c=:blue,label="Polynomial 5th order")
plot!(xGrid,f2.(xGrid),c=:red,label="Linear model")
scatter!(xVals,yVals,c=:black,shape=:xcross,ms=8,label="Data points",
    xlims=(-5,20),ylims=(-50,50),xlabel="x",ylabel="y")                *)
```

Out[1000]= 6

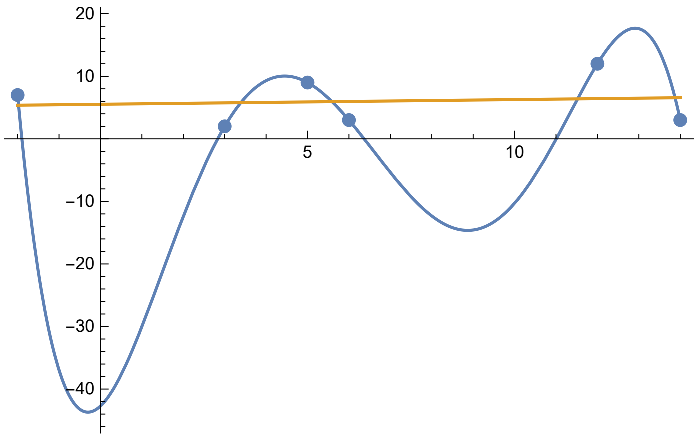
Out[1002]//MatrixForm=

$$\begin{pmatrix} 1 & -2 & 4 & -8 & 16 & -32 \\ 1 & 3 & 9 & 27 & 81 & 243 \\ 1 & 5 & 25 & 125 & 625 & 3125 \\ 1 & 6 & 36 & 216 & 1296 & 7776 \\ 1 & 12 & 144 & 1728 & 20736 & 248832 \\ 1 & 14 & 196 & 2744 & 38416 & 537824 \end{pmatrix}$$

Out[1003]= {-42.7154, 6.19747, 8.96028, -2.72956, 0.260934, -0.00805354}

Out[1004]= $-42.7154 + 6.19747\,x + 8.96028\,x^2 - 2.72956\,x^3 + 0.260934\,x^4 - 0.00805354\,x^5$

Out[1010]=




```
(* In Julia,

A=[ones(n) xVals]
beta0,beta1=pinv(A)*yVals                    *)
```

In[1030]:=

```
(* A random 3 by 3 matrix with entries in {1,2,3}. *)
RandomInteger[{1, 3}, {3, 3}] // MatrixForm
(* The same as a function: *)
rm[] := RandomInteger[{1, 3}, {3, 3}];
rm[]

(* In Julia,

rm()=rand([1,2,3],3,3)                    *)
```

Out[1030]//MatrixForm=

$$\begin{pmatrix} 3 & 1 & 1 \\ 1 & 2 & 3 \\ 3 & 2 & 3 \end{pmatrix}$$

Out[1032]= {{2, 1, 2}, {3, 3, 1}, {2, 1, 3}}

In[1071]:=
```
(* Construct a list of 10^5 such random matrices. *)
mats = Table[rm[], {i, 1, 10^5}];
R = MatrixRank /@ mats;
(* The number of elements of each rank *)
S = Counts[R];
T = Sort[S]
W = Prepend[Table[T[[j]], {j, 1, 3}], 0]
(* 80% of these matrices are invertible. *)
N[(1/10^5) * W]

(* In Julia,
using StatsBase #may need install
mats=[rm() for _ in 1:10^5];
counts(rank.(mats),0:3) *)

(* sum(det.(mats).==0) *)
```

Out[1074]= ⟨| 1 → 273, 2 → 19 885, 3 → 79 842 |⟩

Out[1075]= {0, 273, 19 885, 79 842}

Out[1076]= {0., 0.00273, 0.19885, 0.79842}