```
NB. J BY EXAMPLE
NB.             J is product of JSoftware Inc.  http://jsoftware.com
NB. v3 07/04/2005 (C) Oleg Kobchenko  http://olegykj.sourceforge.net


NB. simple arithmetic ============================================

   2 + 2         NB. comment is 'NB.'
4
   2 - 3         NB. negative numbers use '_'
_1
   2 * 3 + 4     NB. no precedence, right to left
14
   (2 * 3) + 4   NB. parentheses changes order
10
   3 % 4         NB. division represented by '%'
0.75
   *: 4          NB. square
16
   %: 4          NB. square root
2
   % 4           NB. 1/x
0.25
NB. operations using lists =======================================

   2 * 1 2 3     NB. numeric list with space separators
2 4 6
   1 2 3 % 2 4 6 NB. list to list operations, same size
0.5 0.5 0.5
   #1 2 3        NB. size of vector
3
   3$1           NB. generate sequence of same numbers
1 1 1
   5$1 2         NB. or from a list of given elements
1 2 1 2 1
NB. list elements ================================================

   {.1 2 3       NB. first element
1
   {:1 2 3       NB. last element
3
   }.1 2 3       NB. rest without first element
2 3
   }:1 2 3       NB. rest without last element
1 2
   |.1 2 3       NB. reverse
3 2 1
```

```
/K by EXAMPLE
/  K is product of Kx Inc.  http://kx.com
/  2005.06.29. Attila Vrabecz (VrAbi) http://vrabi.web.elte.hu/k
/based on J by EXAMPLE by
/  06/11/2005 (C) Oleg Kobchenko     http://olegykj.sourceforge.net
/simple arithmetic ===============================================

  2+2           /comment is ' /': left of /: whitespace or nothing
4
  2-3           /negative numbers
-1
  2*3+4         /no precedence, right to left
14
  (2*3) + 4     /parentheses changes order
10
  3%4           /division represented by '%'
0.75
  _sqr 4        /square
16.0
  _sqrt 4       /square root
2.0
  %4            /1/x
0.25
/operations using lists ==========================================

  2*1 2 3       /numeric list with space separators
2 4 6
  1 2 3%2 4 6   /list to list operations, same size
0.5 0.5 0.5
  #1 2 3        /size of vector
3
  3#1           /generate sequence of same numbers
1 1 1
  5#1 2         /or from a list of given elements
1 2 1 2 1
/ list elements ==================================================

  *1 2 3        /first element
1
  *|1 2 3       /last element
3
  1_1 2 3       /rest without first element
2 3
  -1_1 2 3      /rest without last element
1 2
  |1 2 3        /reverse
3 2 1
```

```
NB. indexing and sorting ============================================

   1{1 2 3       NB. indexing is zero-based
2
   1 0{1 2 3     NB. index can be vector too
2 1
   i.3           NB. generate zero-based sequence
0 1 2
   2 4 6 i. 4    NB. index of given element(s)
1
   /:2 1 6       NB. indices of sorted order
1 0 2
   /:~2 1 6      NB. sort vector
1 2 6            NB. F~y <=> y F y
NB. list aggregation ================================================

   1 2 3,10 20          NB. join vectors
1 2 3 10 20
   1 + 2 + 3            NB. sum of elements
6
   +/1 2 3              NB. insert '+' between elements
6
   +/\1 2 3             NB. running sum of elements
1 3 6
   1,(1+2),(1+2+3)      NB. same as this
1 3 6
   2+/\1 2 3 4 5        NB. sum or running pairs
3 5 7 9
   _2+/\1 2 3 4 5       NB. non-intersecting pairs
3 7 5
   (<1 2),3 4 6;7 6     NB. < is boxing, ; is box and join
+---+-----+---+
|1 2|3 4 6|7 6|
+---+-----+---+
   >{. 3 4 6;7 6        NB. > is unboxing
3 4 6

NB. function combinations ===========================================

   (+ *:) 4             NB. hook (F G) y <=> y F (G y)
20                      NB.    a + a^2
   (%: , *:) 4          NB. fork (F G H) y <=> (F y) G (H y)
2 16                    NB.    [sqrt(a), a^2]
   *:@(+/) 2 3          NB. composition (F o G) y <=> F G y
25                      NB.    (a + b)^2
   2 +&*: 3             NB. x F & G y <=> (G x) F (G y)
13                      NB.    a^2 + b^2
   2 (+&*: + 2: * *) 3  NB. (a + b)^2 = a^2 + b^2 + 2ab
25                      NB.    0: 1: 2: ... are const functions
   3 +&.*: 4            NB. F&.G y <=> (G^:_1) F G y
5                       NB.    sqrt(a^2 + b^2)
```

```
/ indexing and sorting ============================================

  1 2 3@1       /indexing is zero-based
2
  1 2 3@1 0     /index can be vector too
2 1
  !3            /generate zero-based sequence
0 1 2
  2 4 6?4       /index of given element(s)
1
  <2 1 6        /indices of sorted order
1 0 2
  {x@<x}2 1 6   /sort vector
1 2 6
/ list aggregation ================================================

  1 2 3,10 20     /join vectors
1 2 3 10 20
  1 + 2 + 3       /sum of elements
6
  +/1 2 3         /insert '+' between elements
6
  +\1 2 3         /running sum of elements
1 3 6
  1,(1+2),(1+2+3) /same as this
1 3 6
  +':1 2 3 4 5    /sum running pairs
3 5 7 9
  +/'{(2*!-_-.5*#x)_ x}1 2 3 4 5 /non-intersecting pairs
3 7 5
  (1 2;3 4 6;7 6) /list
(1 2
 3 4 6
 7 6)
  *(3 4 6;7 6)    /first item in the list
3 4 6
/ function combinations ===========================================

  {x+_sqr x}4           /a + a^2
20.0
  (_sqrt;_sqr)@\:4      /[sqrt(a), a^2]
2 16.0
  _sqr+/2 3             /(a +b)^2
25.0
  +/_sqr 2 3            /a^2 + b^2
13.0
  {+/(_sqr x),2*/x}2 3  /(a + b)^2 = a^2 + b^2 + 2ab
25.0
   _sqrt+/_sqr 3 4      /sqrt(a^2 + b^2)
5.0
```

```
NB. user defined functions and arguments ============================

   m1=: -                    NB. ambivalent tacit
   m2=: 3 : '-y.'            NB. monadic explicit
   m3=: 4 : 'x.-y.'          NB. dyadic explcit

     (m1 ,  m2    , 0&m3) 4  NB. monadic use, 0& is bonding
_4 _4 _4
   3 (m1 , (+ m2) ,   m3) 4  NB. dyadic use, hook for dyadization
_1 _1 _1
     (m1 , m3) / 3 4         NB. distribute arguments: dyadization
_1 _1
   3 (m1 , m4) @ , 4         NB. collect arguments: monadization
_3 _4 _3 _4
```

```
/ user defined functions and arguments ============================

  d1:-          /dyadic projection
  d2:{x-y}      /explicit dyad
  m1:-:         /monadic projection
  m2:0-         /monadic projection
  m3:{-x}       /explicit monad

  (m1;m2;m3)@\:4     / monads
-4 -4 -4
  (d1;d2).\:3 4      / dyads
-1 -1
```

```
NB. exponent and logarithm ==========================================

   1x1 2x1 1x2          NB. e, 2e, e squared
2.71828 5.43656 7.38906
   ^2                   NB. exponent, e^2
7.38906
   2^16                 NB. exponent base 2, 2^16
65536
   ^. 1x2               NB. logarithm, ln e^2
2
   2^.65536             NB. logarithm base 2, log2 65536
16
```

```
/ exponent and logarithm ==========================================

  (e;2*e;_sqr e:_exp 1) /e, 2e, e squared
2.718282 5.436564 7.389056
  _exp 2                /exponent, e^2
7.389056
  2^16                  /exponent base 2, 2^16
65536.0
  _log _exp 2           /logarithm, ln e^2
2.0
  _log[65536]%_log[2]   /logarithm base 2, log2 65536
16.0
```

```
NB. trigonometry ====================================================

   1p1 2p1 1p2              NB. pi, 2 pi, pi squared
3.14159 6.28319 9.8696
   load'trig'               NB. load trigonometry library
   cos 1p1                  NB. cosine of pi
_1
   (*:cos 1p1) + *:sin 1p1  NB. theorem of trigonometry
1
   (cos +&*: sin) 1 2p1 1p2 NB. same using fork and &
1 1 1
```

```
/ trigonometry ====================================================

  :a:(pi;2*pi;_sqr pi:_acos-1) /pi, 2 pi, pi squared
3.141593 6.283185 9.869604
  _cos pi                      /cosine of pi
-1.0
  (t:+/_sqr(_cos;_sin)@\:)pi   /theorem of trigonometry
1.0
  t a                          /test theorem at angles
1 1 1.0
```

```
NB. matrices ========================================================

   1 2 3 */ 1 2 3  NB. outer product: multiplication table
1 2 3              NB. same as */~ 1 2 3
2 4 6
3 6 9
   =/~i.3          NB. identity matrix, also =@i. (self-classify)
1 0 0              NB. F~y <=> y F y
0 1 0
0 0 1
   ]M=. i.2 3       NB. generate matrix
0 1 2
3 4 5
   2 2$0 1 1 1     NB. reshape given vector to matrix
0 1
1 1
```

```
/ matrices ========================================================

  1 2 3*/:1 2 3 /outer product: multiplication table
(1 2 3
 2 4 6
 3 6 9)
  {x=/:x}@!3    /identity matrix
(1 0 0
 0 1 0
 0 0 1)
  2 3#!6        /generate matrix
(0 1 2
 3 4 5)
  2 2#0 1 1 1   /reshape given vector to matrix
(0 1
 1 1)
```

```
NB. structural transforms ========================================

   ,N=: i.2 2 3              NB. ravel: list of atoms
0 1 2 3 4 5 6 7 8 9 10 11
   ,"2 N                     NB. ravel each sub-matrix
0 1 2 3  4  5
6 7 8 9 10 11

 (]; |:; |.;|."1;1&|.) M=. 3 3$'ABC123!@#' NB. character matrix
+---+---+---+---+---+---+      NB.  ]    returns argument
|ABC|A1!|!@#|CBA|123|      NB.  |:   transposes
|123|B2@|123|321|!@#|      NB.  |.   reverses outer list
|!@#|C3#|ABC|#@!|ABC|      NB.  |."1 reverses inner list
+---+---+---+---+---+---+      NB. 1|.   rotates outer list

   ;:^:_1 </.M               NB. oblique: secondary diagonals
A B1 C2! 3@ #                NB.   same as (</.~&, +"0/~@i.@#) M
                             NB. ;:^:_1 is inverse of boxing tokens

  i.@# } M                   NB. main diagonal
A2#
```

```
/ structural transforms ========================================

  ,//N:2 2 3#!12             /ravel: list of atoms
0 1 2 3 4 5 6 7 8 9 10 11
  ,/'N                       /ravel each sub-matrix
(0 1 2 3 4 5
6 7 8 9 10 11)
  M:3 3#"ABC123!@#" /character matrix
  (:: ;+: ;|: ;|:'; 1!)@\:M
(("ABC"          / ::  returns argument
  "123"
  "!@#")
 ("A1!"          / +:  transposes
  "B2@"
  "C3#")
 ("!@#"          / |:  reverses items
  "123"
  "ABC")
 ("CBA"          / |:' reverses each items
  "321"
  "#@!")
 ("123"          / 1!  rotates items
  "!@#"
  "ABC"))
  M ./:/:f@=+/'f:,/n,/:\:n:!3  /secondary diagonals
(,"A"
 "B1"
 "C2!"
 "3@"
 ,"#")
  M ./:a,'a:!#M /main diagonal
"A2#"
```

```
NB. selection ====================================================

   1{1{1{N                   NB. repetitive selection of items From list
10
   1{^:3 N                   NB. apply select 3 times
10
   (<1 1 1){N                NB. scatter select
10
   1 1 1 ({~ <)~ N           NB. using unboxed list
10
```

```
/ selection ====================================================

   ((N 1) 1) 1 /repetitive selection of items From list
10
   3@[;1]/N    /apply select 3 times
10
   N[1;1;1]    /scatter select
10
   N . 1 1 1   /scatter select too
10
```

```
NB. factorial and binomial =======================================

   ! 1+i.5          NB. factorial
1 2 6 24 120
   */\ 1+i.5        NB. running product
1 2 6 24 120
   !/~ i.5          NB. binomial coefficients
1 1 1 1 1
0 1 2 3 4
0 0 1 3 6
0 0 0 1 4
0 0 0 0 1
   +/@(! |.)\i. 15  NB. fibonacci: sum of second diagonal of binomial matrix
1 1 2 3 5 8 13 21 34 55 89 144 233 377 610
```

```
/ factorial and binomial =======================================

  (f:{:[x<0;0;*/1.+!x]})'1+!5 /factorial
1 2 6 24 120.0
  *\1+!5                      /running product
1 2 6 24 120
  (b:{(!x){:[x<y;0;_ f[x]%f[y]*f x-y]}\:/:!x})5 /binomial coeff.
(1 1 1 1 1
 0 1 2 3 4
 0 0 1 3 6
 0 0 0 1 4
 0 0 0 0 1)
  1_{+/b[x]./:+(!x;|!x)}'!16 /fibonacci: sum of second diagonal of
                             /binomial matrix
1 1 2 3 5 8 13 21 34 55 89 144 233 377 610
```

```
NB. dot product ====================================================

   1 2 3(+/ . *)1 2 3      NB. dot product
14
   M=: 2 2$0 1 1 1         NB. assignment
   dot=: +/ . *            NB. expression given a name
   dot~ M                  NB. matrix squared
1 1
1 2
   dot^:(15)~ M            NB. matrix to the power of 15, also fibonacci
610  987
987 1597
   {:@{."2 dot^:(<15)~  M  NB. F^:n is apply F n times accumulatively
1 1 2 3 5 8 13 21 34 55 89 144 233 377 610

NB. randomness and probability =====================================

   ]A=: 5 ?@$ 0          NB. 5 random floats from 0..1
0.57708 0.542732 0.488337 0.26004 0.0101683
   ]B=: 10 ?@$ 2         NB. coin toss
1 1 1 0 0 1 1 0 1 0
   ]C=: 3 ? 3            NB. deal 3 out of 3 cards in certain order
1 2 0
   (<./ , >./) A         NB. min and max over the list
0.0101683 0.57708
   B i. 0                NB. first zero
3
   (+/ % #) C-:"1 (?~"0) 10000#3   NB. method monte carlo
0.1637                   NB.    -: is list equality, F"n is rank modifier
   %!3                   NB. exact probability of 3 cards in given order
0.166667
NB. unique elements ================================================

   ]D=.~. S=. 'mississippi'  NB. distinct (nub)
misp
   ]K=. D i. S               NB. key (index)
0 1 2 2 1 2 2 1 3 3 1
   K </. S                   NB. group by key
+-+----+----+--+
|m|iiii|ssss|pp|
+-+----+----+--+
   K #/. S                   NB. frequencies
1 4 4 2

   ]I=. ~: S                 NB. sieve of nub
1 1 1 0 0 0 0 0 1 0 0        NB. where D is in S

   I # S        NB. filter by sieve to get D
misp
   +/"1 =S      NB. = is classify, bool matrix of S vs D
1 4 4 2         NB. where items of D are in S
```

```
/ dot product ======================================================

  1 2 3_dot 1 2 3       /dot product _dot=+/* (optimized)
14
  :M:(0 1;1 1)          /assignment
  :M:(0 1
1 1)                    /equivalent to this
  M _mul M              /matrix squared _mul=_dot\: (optimized)
(1 1
 1 2)
  15_mul[M]/M          /matrix to the power of 15, also fibonacci
(610  987
 987 1597)
  (14_mul[M]\M)[;0;1]
1 1 2 3 5 8 13 21 34 55 89 144 233 377 610
/ randomness and probability =======================================

  :A:5_draw 0          /5 random floats from 0..1
0.03505812 0.7834427 0.7999031 0.9046515 0.2232866
  :B:10_draw 2         /coin toss
1 1 1 0 1 0 1 1 0 0
  :C:3_draw-3          /deal 3 out of 3 cards in certain order
1 0 2
  (&/;|/)@\:A          /min and max over the list
0.03505812 0.9046515
  B?0                  /first zero
3
  {(+/x)%#x}C~/:10000{3_draw-3}\_n  / method monte carlo
0.1643836
  %f 3                 /exact probability of 3 cards in given order
0.1666667
/ unique elements ==================================================

  :D:?S:"mississippi" /?: is unique
"misp"
  :K:D?/:S            /find (?) indexes
0 1 2 2 1 2 2 1 3 3 1
  S@=K                /= is group, group by key
(,"m"
 "iiii"
 "ssss"
 "pp")
  #:'=S               /frequencies
1 4 4 2
  :I:(!#S)_lin*:'=S   /sieve of nub
1 1 1 0 0 0 0 0 1 0 0 /where D is in S
  S@&I                /filter by sieve to get D
"misp"
  +/D=/:S             /where items of D are in S
1 4 4 2
```