```
4821 ( IDEAL Simplified)   EMPTY
4822   ( Static, abstract things)
4823   ( Simple Interactions)
4824 ( Existence Simplified)
4825   ( Benefit of Experience)
4826   ( Stream of Intelligence)
```

arrayForth-3   5/05/25 17:37:52   sF386/NT aF-3d+GLOW FULL In-house

This is the "minimalist" version for x86, with no "hooks" for
  future (unknown) complications. Will make conventions that
  should survive into more complex versions, but this one will
  be a "floor" especially in its native chip version.

Each sort of "thing" will have a numeric handle starting at 1.
  The zero handle will mean "null" or undefined.

BOREDOMLEVEL is bias between self-satisfaction & disengagement.

```
 0 ( IDEAL Simplified)   EMPTY
 1
 2 ( Database)  1 FH 2 FH THRU  ( Exist)  3 FH 5 FH THRU
 3
 4 2 BOREDOMLEVEL !
 5 : .EXP  "EXP TYPE ;   : .RES  "RES TYPE ;
 6 : .INT ( i)  INTS U@ 256 /MOD  SWAP .EXP ." ->" .RES SPACE ;
 7 : .MOOD ( i)  "MOOD TYPE ;
 8
 9 : RUN  E1 PREVEXP !  20 0 DO  STEP  CR I . ." : "
10    CURINT @ DUP . .INT  #SATIS ?
11    EXPECTED @ .RES  SPACE MOOD @ .MOOD  LOOP ;
12
13
14
15
```

Experiences, Results, moods are simple, named abstract things.
  They correspond to processes or events in the Existence
  du jour, as well as to additional pertinent attributes.
  The number and attributes of these are small and Existence-
  dependent, so we denote them internally with small ordinals
  and define only the "null" value (0) here.
Attributes such as names apply to compile-time and to running on
  "big" computers.

M prefix is maximum. # prefix is number of. " is display name.
:ID adds an abstract thing and gives it a programmatic name.
!SARRAY sets display name for the given ordinal and its class.

```
 0   ( Static, abstract things)
 1 : :ID ( a lim _id - i)  OVER @ = ABORT" Too many!"
 2   DUP @  DUP CONSTANT  SWAP TALLY ;
 3 : !SARRAY ( i a _ - i)  SWAP 4* +  HERE SWAP !  32 STRING ;
 4
 5 10 CONSTANT MEXP  10 CONSTANT MRES    6 CONSTANT MMOOD
 6    VARIABLE #EXP     VARIABLE #RES     VARIABLE #MOOD
 7 MEXP SARRAY "EXP  MRES SARRAY "RES  MMOOD SARRAY "MOOD
 8
 9 : :EXP ( _id _dsp - i)  #EXP  MEXP :ID  DUP ['] "EXP !SARRAY ;
10 : :RES ( same)  #RES  MRES :ID  DUP ['] "RES !SARRAY ;
11 : :MOOD ( sam)  #MOOD  MMOOD :ID  DUP ['] "MOOD !SARRAY ;
12
13 :EXP E0 Null DROP  :RES R0 Null DROP  :MOOD MNOTH Nothing DROP
14
15
```

Simple interactions are indexed by an experiment and a result,
  unique in that combination. All are created dynamically by
  the environment. Changes in algorithm may add attributes or
  change storage or indexing methods. Handle is ordinal, which
  simply denotes order of creation. Ordinal 0 is null.

INTS is index; index values res|exp bytes in a halfcell.
?E-R returns nonzero ordinal if the index value exists, index
  value under false if not.
?INT same but takes experiment & result ordinals.
: +INT finds existing or creates new interaction.

```
 0   ( Simple Interactions)
 1 MEXP DUP * CONSTANT MINT  VARIABLE #INT  1 #INT !
 2 MINT HARRAY INTS
 3
 4 : ?E-R ( nE-R - iI | nE-R 0)  #INT @ 1- FOR
 5     DUP I INTS U@ = IF  DROP R> EXIT  THEN NEXT  0 ;
 6 : ?INT ( iE iR - iI | nE-R 0)  >< + ?E-R ;
 7 : +INT ( iE iR - iI)  ?INT  ?DUP 0= IF
 8    #INT @ DUP MINT = ABORT" Too many prim interactions!"
 9    SWAP OVER INTS H!  #INT TALLY  THEN ;
10
11
12
13
14
15
```

Static abstract things are declared at compilation time.

PREVEXP is index of experiment made just before the current.
CUREXP is index of experiment being made by current cycle.

MOOD is an abstract state influencing decision making.
#SATIS is self-satisfaction counter
BOREDOMLEVEL is bias between self-satisfaction & disengagement.

EXPECTED result from current experiment (may be null)
RESULT actual result from current experiment
CURINT interaction for current experiment

```
 0 ( Existence Simplified)
 1 :EXP E1 E1 DROP  :RES R1 R1 DROP
 2 :EXP E2 E2 DROP  :RES R2 R2 DROP
 3
 4 :MOOD SATIS SelfSatisfied DROP    :MOOD FRUST Frustrated DROP
 5 :MOOD BORED Bored DROP            :MOOD PAIN  Pained DROP
 6 :MOOD PLEAS Pleased DROP
 7
 8 ( State variables)
 9 VARIABLE PREVEXP   VARIABLE CUREXP
10 VARIABLE MOOD      VARIABLE #SATIS   VARIABLE BOREDOMLEVEL
11
12 VARIABLE EXPECTED  VARIABLE RESULT  VARIABLE CURINT
13
14
15
```

PREDICT finds the newest interaction employing the given
  experiment (highest ordinal) returning handle for its result.
  If no such interaction exists, returns null result. Because
  the result of each experiment is always the same, this is for
  appearances only.

ENACT performs experiment CUREXP. E1->R1, E2->R2.

OTHEREXP finds the experiment with *highest* ordinal differing
  from the one given. The prototype given returned the *LOWEST*
  ordinal, but no obvious reason why this would be preferable to
  the highest. Second version uses highest instead because that
  covers the case of defaulting to null when there's no other
  experiment than the one being enacted.

```
 0  ( Benefit of Experience)
 1 : PREDICT ( exp - res)  #INT @ 1- FOR  DUP I INTS C@ = IF
 2     DROP R> INTS 1+ C@  EXIT  THEN NEXT  DROP 0 ;
 3
 4 : ENACT ( - res)  CUREXP @ E1 = IF R1  ELSE R2  THEN ;
 5
 6 : OTHEREXP ( cur - new)  #EXP @ 1 DO  DUP I - IF
 7    DROP 2R> NIP  EXIT  THEN LOOP  DROP 0 ;
 8
 9 EXIT
10 : OTHEREXP ( cur - new)  #EXP @ 1- FOR  DUP I - IF
11    DROP  R> EXIT  THEN NEXT  ( Never completes here) ;
12
13
14
15
```

STEP performs one step/epoch of a "stream of intelligence".
  This one is not very bright at all.

```
 0  ( Stream of Intelligence)
 1 : STEP  PREVEXP @  MOOD @ BORED = IF OTHEREXP  0 #SATIS !
 2    THEN DUP CUREXP !  DUP PREDICT EXPECTED !
 3      ENACT DUP RESULT !  +INT CURINT !
 4   RESULT @ EXPECTED @ - IF  FRUST MOOD !  0 #SATIS !
 5    ELSE  SATIS MOOD !  #SATIS TALLY  THEN
 6   #SATIS @ BOREDOMLEVEL @ < NOT IF  BORED MOOD !  THEN
 7   CUREXP @ PREVEXP ! ;
 8
 9
10
11
12
13
14
15
```

```
STEP in 4826 hides 634
0  : 1 E1->R1 0 Null Frustrated
1  : 1 E1->R1 1 R1 SelfSatisfied
2  : 1 E1->R1 2 R1 Bored
3  : 2 E2->R2 0 Null Frustrated
4  : 2 E2->R2 1 R2 SelfSatisfied
5  : 2 E2->R2 2 R2 Bored
6  : 1 E1->R1 1 R1 SelfSatisfied
7  : 1 E1->R1 2 R1 Bored
8  : 2 E2->R2 1 R2 SelfSatisfied
9  : 2 E2->R2 2 R2 Bored
10 : 1 E1->R1 1 R1 SelfSatisfied
11 : 1 E1->R1 2 R1 Bored
12 : 2 E2->R2 1 R2 SelfSatisfied
13 : 2 E2->R2 2 R2 Bored
14 : 1 E1->R1 1 R1 SelfSatisfied
15 : 1 E1->R1 2 R1 Bored
16 : 2 E2->R2 1 R2 SelfSatisfied
17 : 2 E2->R2 2 R2 Bored
18 : 1 E1->R1 1 R1 SelfSatisfied
19 : 1 E1->R1 2 R1 Bored
```