ZEN of the Launchpad

Lessons / Examples eForth 430G2553

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

BLINK.4th

```
 HEX
 : SEC  400 FOR 400 FOR NEXT NEXT ;
 : ON  41 21 C! ;
 : OFF  0 21 C! ;
  : BLINK  BEGIN ON SEC OFF SEC ?KEY UNTIL DROP ;
  BLINK
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

STARS

```
DECIMAL

: STAR 42 EMIT ;

: MARGIN CR 5 SPACES ;

: BALKEN MARGIN STAR STAR STAR STAR STAR ;

: BLIP   MARGIN STAR ;

: F  ( -- )
   BALKEN
   BLIP
   BALKEN
   BLIP
   BLIP
   CR
;

LESSON  1
( Example 1.      The Universal Greeting )

: HELLO CR ." Hello, world!" ;

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

LESSON  2
( Example 2.      The Big F )

: bar   CR ." *****" ;
: post  CR ." *    " ;
: F     bar post bar post post post ;

\ Type 'F' and a return on your keyboard, and you will see a large
\ F character displayed on the screen
```

```
LESSON  3

( Example 3.      FIG, Forth Interest Group )

: center CR ."   *  " ;
: sides  CR ." *   *" ;
: triad1 CR ." * * *" ;
: triad2 CR ." **  *" ;
: triad3 CR ." *  **" ;
: triad4 CR ."  *** " ;
: quart  CR ." ** **" ;
: right  CR ." * ***" ;
: bigT  bar center center center center center center ;
: bigI  center center center center center center center ;
: bigN  sides triad2 triad2 triad1 triad3 triad2 sides ;
: bigG  triad4 sides post right triad1 sides triad4 ;
: FIG  F bigI bigG ;
```

```
LESSON  4

( Example 4.      Repeated Patterns )

\ FOR     [ index -- ]          Set up loop given the index.
\ NEXT    [ -- ]                Decrement index by 1.  If index<0, exit.
\                               If index=limit, exit loop; otherwise
\                               Otherwise repeat after FOR.
\ R@      [ -- index ]          Return the current loop index.

DECIMAL

VARIABLE WIDTH                  ( number of asterisks to print )

: ASTERISKS ( -- , print n asterisks on the screen, n=width )
        WIDTH @                 ( limit=width, initial index=0 )
        FOR ." *"               ( print one asterisk at a time )
        NEXT                    ( repeat n times )
        ;

: RECTANGLE ( height width -- , print a rectangle of asterisks )
        WIDTH !                 ( initialize width to be printed )
        FOR     CR
                ASTERISKS       ( print a line of asterisks )
        NEXT
        ;

: PARALLELOGRAM ( height width -- )
        WIDTH !
        FOR     CR R@ SPACES    ( shift the lines to the right )
                ASTERISKS       ( print one line )
        NEXT
        ;

: TRIANGLE ( width -- , print a triangle area with asterisks )
        FOR     CR
                R@ WIDTH !      ( increase width every line )
                ASTERISKS       ( print one line )
        NEXT
        ;

\ Try the following instructions:

\        3 10 RECTANGLE
\        5 18 PARALLELOGRAM
\        12 TRIANGLE

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
LESSON  5
( Example 5.      The Theory That Jack Built )
( This example shows you how to build a hiararchical structure in Forth)

DECIMAL

: the           ." the " ;
: that          CR ." That " ;
: this          CR ." This is " the ;
: jack          ." Jack Builds" ;
: summary       ." Summary" ;
: flaw          ." Flaw" ;
: mummery       ." Mummery" ;
: k             ." Constant K" ;
: haze          ." Krudite Verbal Haze" ;
: phrase        ." Turn of a Plausible Phrase" ;
: bluff         ." Chaotic Confusion and Bluff" ;
: stuff         ." Cybernatics and Stuff" ;
: theory        ." Theory " jack ;
: button        ." Button to Start the Machine" ;
: child         ." Space Child with Brow Serene" ;
: cybernatics   ." Cybernatics and Stuff" ;

: hiding        CR ." Hiding " the flaw ;
: lay           that ." Lay in " the theory ;
: based         CR ." Based on " the mummery ;
: saved         that ." Saved " the summary ;
: cloak         CR ." Cloaking " k ;
: thick         IF that ELSE CR ." And " THEN
                ." Thickened " the haze ;
: hung          that ." Hung on " the phrase ;
: cover         IF that ." Covered "
                ELSE CR ." To Cover "
                THEN bluff ;
: make          CR ." To Make with " the cybernatics ;
: pushed        CR ." Who Pushed " the button ;
: without       CR ." Without Confusion, Exposing the Bluff" ;

: rest                                  ( pause for user interaction )
       ." . "                           ( print a period )
       10 SPACES                        ( followed by 10 spaces )
       KEY                              ( wait the user to press a key )
       DROP CR CR CR ;

\ Here are new commands needed
\ KEY     [ -- char ]           Wait for a keystroke, and return the
\                               ASCII code of the key pressed.
\ DROP    [ n -- ]              Discard the number.
\ SPACE   [ -- ]                Display a blank.
\ SPACES  [ n -- ]              Display n blanks.
\ IF      [ f -- ]              If the flag is 0, skip the following
\                               instructions up to ELSE or THEN.  If
\                               flag is not 0, execute the following
```

```
\                                   instructions up to ELSE and skip to
\                                   THEN.
\ ELSE    [ -- ]                    Skip the following instructions
\                                   up to THEN.
\ THEN    [ -- ]                    Terminate an IF-ELSE-THEN structure
\                                   or an IF-THEN structure.

: cloaked cloak saved based hiding lay rest ;

: THEORY
       CR this theory rest
       this flaw lay rest
       this mummery hiding lay rest
       this summary based hiding lay rest
       this k saved based hiding lay rest
       this haze cloaked
       this bluff hung 1 thick cloaked
       this stuff 1 cover hung 0 thick cloaked
       this button make 0 cover hung 0 thick cloaked
       this child pushed
              CR ." That Made with " cybernatics without hung
              CR ." And, Shredding " the haze cloak
              CR ." Wrecked " the summary based hiding
              CR ." And Demolished " the theory rest
       ;

( Type THEORY to start)
```

```
LESSON  6

( Example 6.      Help )

( How to use Forth interpreter to carry on a dialog )

: question
        CR CR ." Any more problems you want to solve?"
        CR ." What kind ( sex, job, money, health ) ?"
        CR
        ;

: help  CR
        CR ." Hello!  My name is Creating Computer."
        CR ." Hi there!"
        CR ." Are you enjoying yourself here?"
        KEY 32 OR 89 =
        CR
        IF      CR ." I am glad to hear that."
        ELSE    CR ." I am sorry about that."
                CR ." maybe we can brighten your visit a bit."
        THEN
        CR ." Say!"
        CR ." I can solved all kinds of problems except those dealing"
        CR ." with Greece. "
        question
        ;

: sex   CR CR ." Is your problem TOO MUCH or TOO LITTLE?"
        CR
        ;

: too   ;                                ( noop for syntax smoothness )

: much  CR CR ." You call that a problem?!!  I SHOULD have that problem."
        CR ." If it reall y bothers you, take a cold shower."
        question
        ;

: little
        CR CR ." Why are you here!"
        CR ." You should be in Tokyo or New York of Amsterdam or"
        CR ." some place with some action."
        question
        ;

: health
        CR CR ." My advise to you is:"
        CR ."      1. Take two tablets of aspirin."
        CR ."      2. Drink plenty of fluids."
        CR ."      3. Go to bed (along) ."
        question
        ;
```

```
: job   CR CR ." I can sympathize with you."
        CR ." I have to work very long every day with no pay."
        CR ." My advise to you, is to open a rental computer store."
        question
        ;

: money
        CR CR ." Sorry!  I am broke too."
        CR ." Why don't you sell encyclopedias or marry"
        CR ." someone rich or stop eating, so you won't "
        CR ." need so much money?"
        question
        ;

: HELP help ;
: H help ;
: h help ;

( Type 'help' to start )
```

```
LESSON  7

Example 7.  Money Exchange )

\ The first example we will use to demonstrate how numbers are
\ used in Forth is a money exchange program, which converts money
\ represented in different currencies.  Let's start with the
\ following currency exchange table:

\        30.55 NT        1 Dollar
\        7.73 HK         1 Dollar
\        6.47 RMB        1 Dollar
\        1 Ounce Gold    1285 Dollars
\        1 Ounce Silver  14.95 Dollars

DECIMAL

: NT    ( nNT -- $ )    100 3055 */  ;
: $NT   ( $ -- nNT )    3055 100 */  ;
: RMB   ( nRMB -- $ )   100 647 */  ;
: $RMB  ( $ -- nJmp )   647 100 */  ;
: HK    ( nHK -- $ )    100 773 */  ;
: $HK   ( $ -- $ )      773 100 */  ;
: GOLD  ( nOunce -- $ ) 1285 *  ;
: $GOLD ( $ -- nOunce ) 1285 /  ;
: SILVER ( nOunce -- $ ) 1495 100 */  ;
: $SILVER ( $ -- nOunce ) 100 1495 */  ;
: OUNCE ( n -- n, a word to improve syntax )  ;
: DOLLARS ( n -- )      . ;

\ With this set of money exchange words, we can do some tests:

\        5 ounce gold .
\        10 ounce silver .
\        100 $NT .
\        20 $RMB .

\ If you have many different currency bills in your wallet, you
\ can add then all up in dollars:

\        1000 NT 500 HK + .S
\        320 RMB + .S
\         DOLLARS ( print out total worth in dollars
```

```
LESSON  8

( Example 8. Temperature Conversion )

\ Converting temperature readings between Celcius and Farenheit
\ is also an interesting problem.  The difference between temperature
\ conversion and money exchange is that the two temperature scales
\ have an offset besides the scaling factor.

\ In the following examples, we use these Forth arithmatic operators:

\ +       [ n1 n2 -- n1+n2 ]      Add n1 and n2 and leave sum on stack.
\ -       [ n1 n2 -- n1-n2 ]      Subtract n2 from n1 and leave differrence
\                                 on stack.
\ *       [ n1 n2 -- n1*n2 ]      Multiply n1 and n2 and leave product
\                                 on stack.
\ /       [ n1 n2 -- n1/n2 ]      Divide n1 by n2 and leave quotient on
\                                 stack.
\ */      [ n1 n2 n3 -- n1*n2/n3] Multiply n1 and n2, divide the product
\                                 by n3 and leave quotient on the stack.

: F>C ( nFarenheit -- nCelcius )
        32 -
        10 18 */
        ;

: C>F ( nCelcius -- nFarenheit )
        18 10 */
        32 +
        ;

\ Try these commands

\ 90 F>C .        shows the temperature in a hot summer day and
\ 0 C>F .         shows the temperature in a cold winter night.
```

```
LESSON  9

( Example 9.  Weather Reporting. )

\ IF-ELSE-THEN structure can be nested.

: WEATHER ( nFarenheit -- )
        DUP     55 <
        IF      ." Too cold!" DROP
        ELSE    85 <
                IF      ." About right."
                ELSE    ." Too hot!"
                THEN
        THEN
        ;

\ You can type the following instructions and get some responses from the
\ computer:

\         90 WEATHER Too hot!
\         70 WEATHER About right.
\         32 WEATHER Too cold.
```

```
LESSON  10

( Example 10.  Print the multiplication table )

( More examples on FOR-NEXT loops. )

DECIMAL

: ONEROW ( nRow -- )
        CR
        DUP 3 U.R 3 SPACES
        1 11
        FOR     2DUP *
                4 U.R
                1 +
        NEXT
        DROP ;

: MULTIPLY ( -- )
        CR CR 6 SPACES
        1 11
        FOR     DUP 4 U.R 1 +
        NEXT DROP
        1 11
        FOR     DUP ONEROW 1 +
        NEXT DROP
        ;

( Type MULTIPLY to print the multiplication table )
```

```
LESSON   11

( Example 11.  Calendars )

( Print monthly calendars for any month in years 1950-2128. )
DECIMAL

VARIABLE JULIAN                         ( 0 is 1/1/1950, good until 2050 )
VARIABLE LEAP                           ( 1 for a leap year, 0 otherwise. )
( 1461 CONSTANT 4YEARS                  ( number of days in 4 years )

: YEAR ( YEAR --, compute Julian date and leap year )
        DUP
        1949 - 1461 4 */MOD             ( days since 1/1/1949 )
        365 - JULIAN !                  ( 0 for 1/1/1950 )
        3 =                             ( modulus 3 for a leap year )
        IF 1 ELSE 0 THEN                ( leap year )
        LEAP !
        DUP 2000 =                      ( 2000 is not a leap year )
        IF 0 LEAP ! THEN
      2000 >                      ( adjust due to year 2000 )
      IF ELSE -1 JULIAN +! THEN
        ;

: FIRST ( MONTH -- 1ST, 1st of a month from Jan. 1 )
        DUP 1 =
        IF DROP 0 EXIT THEN             ( 0 for Jan. 1 )
        DUP 2 =
        IF DROP 31 EXIT THEN            ( 31 for Feb. 1 )
        DUP 3 =
        IF DROP 59 LEAP @ + EXIT THEN   ( 59/60 for Mar. 1 )
        4 - 30624 1000 */
        90 + LEAP @ +                   ( Apr. 1 to Dec. 1 )
        ;

: STARS 60 FOR 42 EMIT NEXT ;           ( form the boarder )

: HEADER ( -- )                         ( print title bar )
        CR STARS CR
        ."      SUN     MON     TUE     WED     THU     FRI     SAT"
        CR STARS CR                     ( print weekdays )
        ;

: BLANKS ( MONTH -- )                   ( skip days not in this month )
        FIRST JULIAN @ +                ( Julian date of 1st of month )
        7 MOD 8 * SPACES ;              ( skip colums if not Sunday   )

: DAYS ( MONTH -- )                     ( print days in a month )
        DUP FIRST                       ( days of 1st this month )
        SWAP 1 + FIRST                  ( days of 1st next month )
        OVER - 1 -                      ( loop to print the days )
        1 SWAP                          ( first day count -- )
        FOR  2DUP + 1 -
```

```
                JULIAN @ + 7 MOD        ( which day in the week? )
                IF ELSE CR THEN         ( start a new line if Sunday )
                DUP  8 U.R              ( print day in 8 column field )
                1 +
        NEXT
        2DROP ;                         ( discard 1st day in this month )

: MONTH ( N -- )                        ( print a month calendar )
        HEADER DUP BLANKS               ( print header )
        DAYS CR STARS CR ;              ( print days   )

: JANUARY       YEAR 1 MONTH ;
: FEBRUARY      YEAR 2 MONTH ;
: MARCH         YEAR 3 MONTH ;
: APRIL         YEAR 4 MONTH ;
: MAY           YEAR 5 MONTH ;
: JUNE          YEAR 6 MONTH ;
: JULY          YEAR 7 MONTH ;
: AUGUST        YEAR 8 MONTH ;
: SEPTEMBER     YEAR 9 MONTH ;
: OCTOBER       YEAR 10 MONTH ;
: NOVEMBER      YEAR 11 MONTH ;
: DECEMBER      YEAR 12 MONTH ;

\ To print the calender of April 1999, type:
\        2011 APRIL
```

```
LESSON  12

( Example 12.      Sines and Cosines )

\ Sines and cosines of angles are among the most often encountered
\ transdential functions, useful in drawing circles and many other
\ different applications.  They are usually computed using floating
\ numbers for accuracy and dynamic range.  However, for graphics
\ applications in digital systems, single integers in the range from
\ -32768 to 32767 are sufficient for most purposes.  We shall
\ study the computation of sines and cosines using the single
\ integers.

\ The value of sine or cosine of an angle lies between -1.0 and +1.0.
\ We choose to use the integer 10000 in decimal to represent 1.0
\ in the computation so that the sines and cosines can be represented
\ with enough precision for most applications.  Pi is therefore
\ 31416, and 90 degree angle is represented by 15708.  Angles
\ are first reduced in to the range from -90 to +90 degrees,
\ and then converted to radians in the ranges from -15708 to
\ +15708.  From the radians we compute the values of sine and
\ cosine.

\ The sines and cosines thus computed are accurate to 1 part in
\ 10000.  This algorithm was first published by John Bumgarner
\ in Forth Dimensions, Volume IV, No. 1, p. 7.

DECIMAL
31415 CONSTANT PI
10000 CONSTANT 10K

VARIABLE XS                             ( square of scaled angle )

: KN ( n1 n2 -- n3, n3=10000-n1*x*x/n2 where x is the angle )
        XS @ SWAP /                     ( x*x/n2 )
        NEGATE 10K */                   ( -n1*x*x/n2 )
        10K +                           ( 10000-n1*x*x/n2 )
        ;

: (SIN) ( x -- sine*10K, x in radian*10K )
        DUP DUP 10K */                  ( x*x scaled by 10K )
        XS !                            ( save it in XS )
        10K 72 KN                       ( last term )
        42 KN 20 KN 6 KN                ( terms 3, 2, and 1 )
        10K */                          ( times x )
        ;

: (COS) ( x -- cosine*10K, x in radian*10K )
        DUP 10K */ XS !                 ( compute and save x*x )
        10K 56 KN 30 KN 12 KN 2 KN      ( serial expansion )
        ;

: SIN ( degree -- sine*10K )
```

```
        PI 180 */                       ( convert to radian )
        (SIN)                           ( compute sine )
        ;

: COS ( degree -- cosine*10K )
        PI 180 */
        (COS)
        ;

\ To test the routines, type:

\        90 SIN .                       10000
\        45 SIN .                        7070
\        30 SIN .                        4999
\         0 SIN .                           0
\        90 COS .                           0
\        45 COS .                        7072
\         0 COS .                        10000
```

LESSON 13

```
( Example 13.      Square Root )

\ There are many ways to take the square root of an integer.  The
\ special routine here was first discovered by Wil Baden.  Wil
\ used this routine as a programming challenge while attending
\ a FORML Conference in Taiwan, 1984.

\ This algorithm is based on the fact that the square of n+1 is equal
\ to the sum of the square of n plus 2n+1.  You start with an 0 on
\ the stack and add to it 1, 3, 5, 7, etc., until the sum is greater
\ than the integer you wished to take the root.  That number when
\ you stopped is the square root.

: SQRT ( n -- root )
        65025 OVER U<                   ( largest square it can handle)
        IF DROP 255 EXIT THEN           ( safety exit )
        >R                              ( save sqaure )
        1 1                             ( initial square and root )
        BEGIN                           ( set n1 as the limit )
                OVER R@ U<              ( next square )
        WHILE
                DUP 2 * 1 +           ( n*n+2n+1 )
                ROT + SWAP
                1 +                      ( n+1 )
        REPEAT
        SWAP DROP
        R> DROP
        ;
```

```
LESSON  14

( Example 14.      Radix for Number Conversions )

DECIMAL

( : DECIMAL       10 BASE ! ; )
( : HEX           16 BASE ! ; )
: OCTAL         8 BASE !  ;
: BINARY        2 BASE !  ;

\ Try converting numbers among different radices:

\        DECIMAL 12345 HEX U.
\        HEX ABCD DECIMAL U.
\        DECIMAL 100 BINARY U.
\        BINARY 101010101010 DECIMAL U.

\ Real programmers impress on novices by carrying a HP calculator
\ which can convert numbers between decimal and hexadecimal.  A
\ Forth computer has this calculator built in, besides other functions.
```

LESSON 15

Example 15. ASCII Character Table)

```
DECIMAL

: CHARACTER ( n -- )
        DUP EMIT HEX DUP 3 .R
        OCTAL DUP 4 .R
        DECIMAL 3 .R
        2 SPACES
        ;

: LINE ( n -- )
        CR
        5 FOR   DUP CHARACTER
                16 +
        NEXT
        DROP ;

: TABLE ( -- )
        32
        15 FOR  DUP LINE
                1 +
        NEXT
        DROP ;
```

```
LESSON  16

( Example 16.  Random Numbers )

\ Random numbers are often used in computer simulations and computer
\ games.  This random number generator was published in Leo Brodie's
\ 'Starting Forth'.

DECIMAL
VARIABLE RND                            ( seed )

: RANDOM ( -- n, a random number within 0 to 65536 )
        RND @ 31421 *                   ( RND*31421 )
        6927 +                          ( RND*31421+6926, mod 65536)
        DUP RND !                       ( refresh he seed )
        ;

: CHOOSE ( n1 -- n2, a random number within 0 to n1 )
        RANDOM UM*                      ( n1*random to a double product)
        SWAP DROP                       ( discard lower part )
        ;                                ( in fact divide by 65536 )

HERE RND !

\ To test the routine, type

\        100 CHOOSE .
\        100 CHOOSE .
\        100 CHOOSE .

\ and varify that the results are randomly distributed betweem 0 and
\ 99 .

HERE RND !                              ( initialize seed )
```

```
LESSON  17

( Example 17.      Guess a Number )

( Require CHOOSE from Lesson16.txt )

DECIMAL

: GetNumber ( -- n )
        BEGIN
                CR ." Enter a Number: " ( show message )
                QUERY BL WORD NUMBER?   ( get a string )
        UNTIL                           ( repeat until a valid number )
        ;

( With this utility instruction, we can write a game 'Guess a Number.' )

: InitialNumber ( -- n , set up a number for the player to guess )
        CR CR CR ." What limit do you want?"
        GetNumber                       ( ask the user to enter a number )
        CR ." I have a number between 0 and " DUP .
        CR ." Now you try to guess what it is."
        CR
        CHOOSE                          ( choose a random number )
        ;                               ( between 0 and limit )

: Check ( n1 -- , allow player to guess, exit when the guess is correct )
        BEGIN   CR ." Please enter your guess."
                GetNumber
                2DUP =                  ( equal? )
                IF      2DROP           ( discard both numbers )
                        CR ." Correct!!!"
                        EXIT
                THEN
                OVER <
                IF      CR ." Too low."
                ELSE    CR ." Too high!"
                THEN    CR
        0 UNTIL                         ( always repeat )
        ;

: Greet ( -- )
        CR CR CR ." GUESS A NUMBER"
        CR ." This is a number guessing game.  I'll think"
        CR ." of a number between 0 and any limit you want."
        CR ." (It should be smaller than 32000.)"
        CR ." Then you have to guess what it is."
        ;

: GUESS ( -- , the game )
        Greet
        BEGIN   InitialNumber                   ( set initial number)
                Check                           ( let player guess )
                CR CR ." Do you want to play again? (Y/N) "
```

```
                KEY                                 ( get one key )
                32 OR 110 =                         ( exit if it is N or n )
        UNTIL
        CR CR ." Thank you.  Have a good day."  ( sign off )
        CR
        ;


\ Type 'GUESS' will initialize the game and the computer will entertain
\ a user for a while.  Note the use of the indefinite loop structure:

\        BEGIN <repeat-clause> [ f ] UNTIL

\ You can jump out of the infinite loop by the instruction EXIT, which
\ skips all the instructions in a Forth definition up to ';', which
\ terminates this definition and continues to the next definition. )
```

```
\ Morse Code on the LaunchPad.
\ Connections:   P2.0 --->------8 Ohm Speaker-----<---GND

DECIMAL

VARIABLE /freq
VARIABLE duration

: init          ( -- )
  1 $2A C! 1 $2E C!   ( P2.0 connected to TA1.0)
  $80 $182 !  ( TA1.0 toggle mode)
  $1000 $192 ! ( TA1CCR0, period )
  500 /freq !  100 duration ! ; ( init them, or app save fails!)
: tone $210 $180 ! ; ( SMCLK, count up )
: stop ( -- )   0 $180 ! 0 $21 C! ;  ( stop tone and LEDs)
: pause       ( d -- )    FOR /freq @ FOR NEXT NEXT ;
: red 1 $21 C! ;
: green $40 $21 C! ;
: short duration @ pause ;
: long  duration @ 3 * pause ;

: dit  init red tone short stop short ;
: dah  init green tone long stop short ;
: ..   stop long ;   ( inter-element gap between the dots and dashes )

\ Morse Alphabet
: A dit dah .. ;
: B dah dit dit dit .. ;
: C dah dit dah dit .. ;
: D dah dit dit .. ;
: E dit .. ;
: F dit dit dah dit .. ;
: G dah dah dit .. ;
: H dit dit dit dit .. ;
: I dit dit ;
: J dit dah dah dah .. ;
: K dah dit dah .. ;
: L dit dah dit dit .. ;
: M dah dah .. ;
: N dah dit .. ;
: O dah dah dah .. ;
: P dit dah dah dit .. ;
: Q dah dah dit dah .. ;
: R dit dah dit .. ;
: S dit dit dit  .. ;
: T dah .. ;
: U dit dit dah .. ;
: V dit dit dit dah .. ;
: W dit dah dah .. ;
: X dah dit dit dah .. ;
: Y dah dit dah dah .. ;
: Z dah dah dit dit .. ;
```

```
: _0 dah dah dah dah dah .. ;
: _1 dit dah dah dah dah .. ;
: _2 dit dit dah dah dah .. ;
: _3 dit dit dit dah dah .. ;
: _4 dit dit dit dit dah .. ;
: _5 dit dit dit dit dit .. ;
: _6 dah dit dit dit dit .. ;
: _7 dah dah dit dit dit .. ;
: _8 dah dah dah dit dit .. ;
: _9 dah dah dah dah dit .. ;

: //  stop 7 FOR short NEXT ; ( Pause between words)

\ Commonly used two letter procedural signals

: AA   A A ;     ( End Of Line)
: AAA  A A A ;   ( Full Stop)
: AR   A R ;     ( End of message)
: AS   A S ;     ( Stand by; wait)
: BK   B K ;     ( Break )
: BT   B T ;     ( Separation - break - between address and text; text +
signature)
: CL   C L ;     ( Going off the air: clear)
: CQ   C .. Q ;  ( Calling any amateur radio station)
: DE   D E ;     ( This or From)
: GB   G B ;     ( Good bye, God Bless)
: GD   G D ;     ( Good, Good Day)
: GE   G E ;     ( Good Evening )
: HH   H H ;     ( Error sending. 8 dits, Transm. cont., last word correctly
sent.)
: II   I I ;     ( Short form of above <HH> )
: IMI  I M I ;   ( Repeat; I say again. Difficult or unusual words or groups.)
: KA   K A ;     ( Beginning of message)
: KN   K N ;     ( Go only, invite a specific station to transmit)
: NR   N R ;     ( Number follows )
: OK   O K ;     ( Correct)
: SGD  S G D ;   ( Signed)
: SK   S K ;     ( Out; clear - end of communications, no reply expected.)
: SOS  dit dit dit dah dah dah dit dit dit .. ;
( Mayday! Without character pauses!)
: VE   V E ;     ( Understood)


: >>  F O R T H  // I S // S U P E R  ;

: TITANIC  BEGIN SOS // ?KEY UNTIL DROP ;

: ZEN
   Z E N // F O R // _4 _3 _0 // L A U N C H P A D // AR
;

( finis)
```

```
LESSONS  SUMMARY

( Example 1.      The Universal Greeting ) \ ============================

DECIMAL

: HELLO CR ." Hello, world!" ;

( Example 2.      The Big F ) \ ========================================

: bar   CR ." *****" ;
: post  CR ." *    " ;
: F     bar post bar post post post ;

( Type 'F' and a return on your keyboard, and you will see a large )
( F character displayed on the screen )

( Example 3.      FIG, Forth Interest Group ) \ ========================

: center CR ."   *  " ;
: sides  CR ." *   *" ;
: triad1 CR ." * * *" ;
: triad2 CR ." **  *" ;
: triad3 CR ." *  **" ;
: triad4 CR ."  *** " ;
: quart  CR ." ** **" ;
: right  CR ." * ***" ;
: bigT  bar center center center center center center ;
: bigI  center center center center center center center ;
: bigN  sides triad2 triad2 triad1 triad3 triad2 sides ;
: bigG  triad4 sides post right triad1 sides triad4 ;
: FIG  F bigI bigG ;


( Example 4.      Repeated Patterns ) \ ================================

FOR     [ index -- ]            Set up loop given the index.
NEXT    [ -- ]                  Decrement index by 1.  If index<0,
exit.                          If index=limit, exit loop; otherwise
                                Otherwise repeat after FOR.
R@      [ -- index ]            Return the current loop index. )


VARIABLE WIDTH                  ( number of asterisks to print )

: ASTERISKS ( -- , print n asterisks on the screen, n=width )
        WIDTH @                 ( limit=width, initial index=0 )
        FOR ." *"               ( print one asterisk at a time )
        NEXT                    ( repeat n times )
        ;

: RECTANGLE ( height width -- , print a rectangle of asterisks )
        WIDTH !                 ( initialize width to be printed )
```

```
        FOR     CR
                ASTERISKS       ( print a line of asterisks )
        NEXT
        ;

: PARALLELOGRAM ( height width -- )
        WIDTH !
        FOR     CR R@ SPACES    ( shift the lines to the right )
                ASTERISKS       ( print one line )
        NEXT
        ;

: TRIANGLE ( width -- , print a triangle area with asterisks )
        FOR     CR
                R@ WIDTH !      ( increase width every line )
                ASTERISKS       ( print one line )
        NEXT
        ;

( Try the following instructions

        3 10 RECTANGLE
        5 18 PARALLELOGRAM
        12 TRIANGLE  )


( Example 5.      The Theory That Jack Built ) \ ==========================
( This example shows you how to build a hiararchical structure in Forth)

DECIMAL

: the           ." the " ;
: that          CR ." That " ;
: this          CR ." This is " the ;
: jack          ." Jack Builds" ;
: summary       ." Summary" ;
: flaw          ." Flaw" ;
: mummery       ." Mummery" ;
: k             ." Constant K" ;
: haze          ." Krudite Verbal Haze" ;
: phrase        ." Turn of a Plausible Phrase" ;
: bluff         ." Chaotic Confusion and Bluff" ;
: stuff         ." Cybernatics and Stuff" ;
: theory        ." Theory " jack ;
: button        ." Button to Start the Machine" ;
: child         ." Space Child with Brow Serene" ;
: cybernatics   ." Cybernatics and Stuff" ;

: hiding        CR ." Hiding " the flaw ;
: lay           that ." Lay in " the theory ;
: based         CR ." Based on " the mummery ;
: saved         that ." Saved " the summary ;
: cloak         CR ." Cloaking " k ;
: thick         IF that ELSE CR ." And " THEN
```

```
                ." Thickened " the haze ;
: hung          that ." Hung on " the phrase ;
: cover         IF that ." Covered "
                ELSE CR ." To Cover "
                THEN bluff ;
: make          CR ." To Make with " the cybernatics ;
: pushed        CR ." Who Pushed " button ;
: without       CR ." Without Confusion, Exposing the Bluff" ;

: rest                                  ( pause for user interaction )
        ." . "                          ( print a period )
        10 SPACES                       ( followed by 10 spaces )
        KEY                             ( wait the user to press a key )
        DROP CR CR CR ;

(
KEY     [ -- char ]             Wait for a keystroke, and return the
                                ASCII code of the key pressed.
DROP    [ n -- ]                Discard the number.
SPACE   [ -- ]                  Display a blank.
SPACES  [ n -- ]                Display n blanks.
IF      [ f -- ]                If the flag is 0, skip the following
                                instructions up to ELSE or THEN.  If
                                flag is not 0, execute the following
                                instructions up to ELSE and skip to
                                THEN.
ELSE    [ -- ]                  Skip the following instructions
                                up to THEN.
THEN    [ -- ]                  Terminate an IF-ELSE-THEN structure
                                or an IF-THEN structure.
)

: cloaked cloak saved based hiding lay rest ;

: THEORY
        CR this theory rest
        this flaw lay rest
        this mummery hiding lay rest
        this summary based hiding lay rest
        this k saved based hiding lay rest
        this haze cloaked
        this bluff hung 1 thick cloaked
        this stuff 1 cover hung 0 thick cloaked
        this button make 0 cover hung 0 thick cloaked
        this child pushed
                CR ." That Made with " cybernatics without hung
                CR ." And, Shredding " the haze cloak
                CR ." Wrecked " the summary based hiding
                CR ." And Demolished " the theory rest
        ;

( Type THEORY to start)
```

```
( Example 6.      Help ) \ ================================================
( How to use Forth interpreter to carry on a dialog )

: question
        CR CR ." Any more problems you want to solve?"
        CR ." What kind ( sex, job, money, health ) ?"
        CR
        ;

: help  CR
        CR ." Hello!  My name is Creating Computer."
        CR ." Hi there!"
        CR ." Are you enjoying yourself here?"
        KEY 32 OR 89 =
        CR
        IF      CR ." I am glad to hear that."
        ELSE    CR ." I am sorry about that."
                CR ." maybe we can brighten your visit a bit."
        THEN
        CR ." Say!"
        CR ." I can solved all kinds of problems except those dealing"
        CR ." with Greece. "
        question
        ;

: sex   CR CR ." Is your problem TOO MUCH or TOO LITTLE?"
        CR
        ;

: too   ;                                ( noop for syntax smoothness )

: much  CR CR ." You call that a problem?!!  I SHOULD have that problem."
        CR ." If it reall y bothers you, take a cold shower."
        question
        ;

: little
        CR CR ." Why are you here!"
        CR ." You should be in Tokyo or New York of Amsterdam or"
        CR ." some place with some action."
        question
        ;

: health
        CR CR ." My advise to you is:"
        CR ."      1. Take two tablets of aspirin."
        CR ."      2. Drink plenty of fluids."
        CR ."      3. Go to bed (along) ."
        question
        ;

: job   CR CR ." I can sympathize with you."
        CR ." I have to work very long every day with no pay."
```

```
        CR ." My advise to you, is to open a rental computer store."
        question
        ;

: money
        CR CR ." Sorry!  I am broke too."
        CR ." Why don't you sell encyclopedias of marry"
        CR ." someone rich or stop eating, so you won't "
        CR ." need so much money?"
        question
        ;

: HELP help ;
: H help ;
: h help ;

( Type 'help' to start )


( Example 7.  Money Exchange  ============================================

The first example we will use to demonstrate how numbers are
used in Forth is a money exchange program, which converts money
represented in different currencies.  Let's start with the
following currency exchange table:

        33.55 NT        1 Dollar
        7.73 HK         1 Dollar
        9.47 RMB        1 Dollar
        1 Ounce Gold    285 Dollars
        1 Ounce Silver  4.95 Dollars )

DECIMAL

: NT    ( nNT -- $ )    100 3355 */  ;
: $NT   ( $ -- nNT )    3355 100 */  ;
: RMB   ( nRMB -- $ )   100 947 */  ;
: $RMB  ( $ -- nJmp )   947 100 */  ;
: HK    ( nHK -- $ )    100 773 */  ;
: $HK   ( $ -- $ )      773 100 */  ;
: GOLD  ( nOunce -- $ ) 285 *  ;
: $GOLD ( $ -- nOunce ) 285 /  ;
: SILVER ( nOunce -- $ ) 495 100 */  ;
: $SILVER ( $ -- nOunce ) 100 495 */  ;
: OUNCE ( n -- n, a word to improve syntax )  ;
: DOLLARS ( n -- )      . ;

( With this set of money exchange words, we can do some tests:

        5 ounce gold .
        10 ounce silver .
        100 $NT .
        20 $RMB .
```

```
If you have many different currency bills in your wallet, you
can add then all up in dollars:

        1000 NT 500 HK + .S
        320 RMB + .S
        DOLLARS ( print out total worth in dollars )


( Example 8. Temperature Conversion  ====================================

Converting temperature readings between Celcius and Farenheit
is also an interesting problem.  The difference between temperature
conversion and money exchange is that the two temperature scales
have an offset besides the scaling factor. )

: F>C ( nFarenheit -- nCelcius )
        32 -
        10 18 */
        ;

: C>F ( nCelcius -- nFarenheit )
        18 10 */
        32 +
        ;

( Try these commands

90 F>C .        shows the temperature in a hot summer day and
0 C>F .         shows the temperature in a cold winter night.

In the above examples, we use the following Forth arithmatic
operators:

+       [ n1 n2 -- n1+n2 ]      Add n1 and n2 and leave sum on stack.
-       [ n1 n2 -- n1-n2 ]      Subtract n2 from n1 and leave differrence
                                on stack.
*       [ n1 n2 -- n1*n2 ]      Multiply n1 and n2 and leave product
                                on stack.
/       [ n1 n2 -- n1/n2 ]      Divide n1 by n2 and leave quotient on
                                stack.
*/      [ n1 n2 n3 -- n1*n2/n3] Multiply n1 and n2, divide the product
                                by n3 and leave quotient on the stack.
.S      [ ... -- ... ]          Show the topmost 4 numbers on stack.
)


( Example 9.  Weather Reporting. )  \ ====================================

: WEATHER ( nFarenheit -- )
        DUP     55 <
        IF      ." Too cold!" DROP
        ELSE    85 <
                IF      ." About right."
                ELSE    ." Too hot!"
```

```
                THEN
        THEN
        ;


( You can type the following instructions and get some responses from the
computer:

        90 WEATHER Too hot!
        70 WEATHER About right.
        32 WEATHER Too cold.
)



( Example 10.  Print the multiplication table ) \ ========================

: ONEROW ( nRow -- )
        CR
        DUP 3 .R 3 SPACES
        1 11
        FOR     2DUP *
                4 .R
                1 +
        NEXT
        DROP ;

: MULTIPLY ( -- )
        CR CR 6 SPACES
        1 11
        FOR     DUP 4 .R 1 +
        NEXT DROP
        1 11
        FOR     DUP ONEROW 1 +
        NEXT DROP
        ;

( Type MULTIPLY to print the multiplication table )


( Example 11.  Calendars ) \ ============================================

( Print weekly calendars for any month in any year. )
DECIMAL

VARIABLE JULIAN                         ( 0 is 1/1/1950, good until 2050 )
VARIABLE LEAP                           ( 1 for a leap year, 0 otherwise. )
( 1461 CONSTANT 4YEARS                  ( number of days in 4 years )

: YEAR ( YEAR --, compute Julian date and leap year )
        DUP
        1949 - 1461 4 */MOD             ( days since 1/1/1949 )
        365 - JULIAN !                  ( 0 for 1/1/1950 )
        3 =                             ( modulus 3 for a leap year )
```

```
        IF 1 ELSE 0 THEN                ( leap year )
        LEAP !
        DUP 2000 =                      ( 2000 is not a leap year )
        IF 0 LEAP ! THEN
        2001 <                          ( correction due to 2000 )
        IF ELSE -1 JULIAN +! THEN
        ;

: FIRST ( MONTH -- 1ST, 1st of a month from Jan. 1 )
        DUP 1 =
        IF DROP 0 EXIT THEN             ( 0 for Jan. 1 )
        DUP 2 =
        IF DROP 31 EXIT THEN            ( 31 for Feb. 1 )
        DUP 3 =
        IF DROP 59 LEAP @ + EXIT THEN   ( 59/60 for Mar. 1 )
        4 - 30624 1000 */
        90 + LEAP @ +                   ( Apr. 1 to Dec. 1 )
        ;

: STARS 60 FOR 42 EMIT NEXT ;           ( form the boarder )

: HEADER ( -- )                         ( print title bar )
        CR STARS CR
        ."       SUN     MON     TUE     WED     THU     FRI     SAT"
        CR STARS CR                     ( print weekdays )
        ;

: BLANKS ( MONTH -- )                   ( skip days not in this month )
        FIRST JULIAN @ +                ( Julian date of 1st of month )
        7 MOD 8 * SPACES ;              ( skip colums if not Sunday   )

: DAYS ( MONTH -- )                     ( print days in a month )
        DUP FIRST                       ( days of 1st this month )
        SWAP 1 + FIRST                  ( days of 1st next month )
        OVER - 1 -                      ( loop to print the days )
        1 SWAP                          ( first day count -- )
        FOR  2DUP + 1 -
                JULIAN @ + 7 MOD        ( which day in the week? )
                IF ELSE CR THEN         ( start a new line if Sunday )
                DUP  8 U.R              ( print day in 8 column field )
                1 +
        NEXT
        2DROP ;                         ( discard 1st day in this month )

: MONTH ( N -- )                        ( print a month calendar )
        HEADER DUP BLANKS               ( print header )
        DAYS CR STARS CR ;              ( print days   )

: JANUARY       YEAR 1 MONTH ;
: FEBRUARY      YEAR 2 MONTH ;
: MARCH         YEAR 3 MONTH ;
: APRIL         YEAR 4 MONTH ;
: MAY           YEAR 5 MONTH ;
: JUNE          YEAR 6 MONTH ;
```

```
: JULY          YEAR 7 MONTH ;
: AUGUST        YEAR 8 MONTH ;
: SEPTEMBER     YEAR 9 MONTH ;
: OCTOBER       YEAR 10 MONTH ;
: NOVEMBER      YEAR 11 MONTH ;
: DECEMBER      YEAR 12 MONTH ;

( To print the calender of April 1999, type:
        1999 APRIL
)



( Example 12.      Sines and Cosines  ====================================

Sines and cosines of angles are among the most often encountered
transdential functions, useful in drawing circles and many other
different applications.  They are usually computed using floating
numbers for accuracy and dynamic range.  However, for graphics
applications in digital systems, single integers in the range from
-32768 to 32767 are sufficient for most purposes.  We shall
study the computation of sines and cosines using the single
integers.

The value of sine or cosine of an angle lies between -1.0 and +1.0.
We choose to use the integer 10000 in decimal to represent 1.0
in the computation so that the sines and cosines can be represented
with enough precision for most applications.  Pi is therefore
31416, and 90 degree angle is represented by 15708.  Angles
are first reduced in to the range from -90 to +90 degrees,
and then converted to radians in the ranges from -15708 to
+15708.  From the radians we compute the values of sine and
cosine.

The sines and cosines thus computed are accurate to 1 part in
10000.  This algorithm was first published by John Bumgarner
in Forth Dimensions, Volume IV, No. 1, p. 7.

31415 CONSTANT PI
10000 CONSTANT 10K )
VARIABLE XS                             ( square of scaled angle )

: KN ( n1 n2 -- n3, n3=10000-n1*x*x/n2 where x is the angle )
        XS @ SWAP /                     ( x*x/n2 )
        10000 */ NEGATE                   ( -n1*x*x/n2 )
        10000 +                           ( 10000-n1*x*x/n2 )
        ;

: (SIN) ( x -- sine*10K, x in radian*10K )
        DUP DUP 10000 */                  ( x*x scaled by 10K )
        XS !                            ( save it in XS )
        10000 72 KN                       ( last term )
        42 KN 20 KN 6 KN                ( terms 3, 2, and 1 )
        10000 */                          ( times x )
```

```
        ;

: (COS) ( x -- cosine*10K, x in radian*10K )
        DUP 10000 */ XS !                  ( compute and save x*x )
        10000 56 KN 30 KN 12 KN 2 KN       ( serial expansion )
        ;

: SIN ( degree -- sine*10K )
        31415 180 */                        ( convert to radian )
        (SIN)                             ( compute sine )
        ;

: COS ( degree -- cosine*10K )
        31415 180 */
        (COS)
        ;


( To test the routines, type:

        90 SIN .                           9999
        45 SIN .                           7070
        30 SIN .                           5000
         0 SIN .                              0
        90 COS .                              0
        45 COS .                           7071
         0 COS .                           10000 )



( Example 13.      Square Root  ========================================

There are many ways to take the square root of an integer.  The
special routine here was first discovered by Wil Baden.  Wil
used this routine as a programming challenge while attending
a FORML Conference in Taiwan, 1984.

This algorithm is based on the fact that the square of n+1 is equal
to the sum of the square of n plus 2n+1.  You start with an 0 on
the stack and add to it 1, 3, 5, 7, etc., until the sum is greater
than the integer you wished to take the root.  That number when
you stopped is the square root.
)

: SQRT ( n -- root )
        65025 OVER U<                     ( largest square it can handle)
        IF DROP 255 EXIT THEN             ( safety exit )
        >R                                ( save sqaure )
        1 1                               ( initial square and root )
        BEGIN                             ( set n1 as the limit )
                OVER R@ U<                ( next square )
        WHILE
                DUP CELLS 1 +             ( n*n+2n+1 )
                ROT + SWAP
```

```
        1 +                         ( n+1 )
REPEAT
SWAP DROP
R> DROP
;
```

```
( Example 14.      Radix for Number Conversions )    \ ======================

DECIMAL

( : DECIMAL       10 BASE ! ; )
( : HEX           16 BASE ! ; )
: OCTAL         8 BASE !  ;
: BINARY        2 BASE !  ;

( Try converting numbers among different radices:

        DECIMAL 12345 HEX U.
        HEX ABCD DECIMAL U.
        DECIMAL 100 BINARY U.
        BINARY 101010101010 DECIMAL U.

Real programmers impress on novices by carrying a HP calculator
which can convert numbers between decimal and hexadecimal.  A
Forth computer has this calculator built in, besides other functions.
)



( Example 15.      ASCII Character Table )  \ ============================

: CHARACTER ( n -- )
        DUP EMIT HEX DUP 3 .R
        OCTAL DUP 4 .R
        DECIMAL 3 .R
        2 SPACES
        ;

: LINE ( n -- )
        CR
        5 FOR   DUP CHARACTER
                16 +
        NEXT
        DROP ;

: TABLE ( -- )
        32
        15 FOR  DUP LINE
                1 +
        NEXT
        DROP ;
```

```
( Example 16.  Random Numbers  ==========================================

Random numbers are often used in computer simulations and computer
games.  This random number generator was published in Leo Brodie's
'Starting Forth'.
)

VARIABLE RND                            ( seed )
HERE RND !                              ( initialize seed )

: RANDOM ( -- n, a random number within 0 to 65536 )
        RND @ 31421 *                   ( RND*31421 )
        6927 +                          ( RND*31421+6926, mod 65536)
        DUP RND !                       ( refresh he seed )
        ;

: CHOOSE ( n1 -- n2, a random number within 0 to n1 )
        RANDOM UM*                      ( n1*random to a double product)
        SWAP DROP                       ( discard lower part )
        ;                                ( in fact divide by 65536 )

( To test the routine, type

        100 CHOOSE .
        100 CHOOSE .
        100 CHOOSE .

and varify that the results are randomly distributed betweem 0 and
99 . )


( Example 17.      Guess a Number )  \ ===================================

: GetNumber ( -- n )
        BEGIN
                CR ." Enter a Number: " ( show message )
                QUERY BL WORD NUMBER?   ( get a string )
        UNTIL                           ( repeat until a valid number )
        ;

( With this utility instruction, we can write a game 'Guess a Number.' )

: InitialNumber ( -- n , set up a number for the player to guess )
        CR CR CR ." What limit do you want?"
        GetNumber                       ( ask the user to enter a number )
        CR ." I have a number between 0 and " DUP .
        CR ." Now you try to guess what it is."
        CR
        CHOOSE                          ( choose a random number )
        ;                               ( between 0 and limit )

: Check ( n1 -- , allow player to guess, exit when the guess is correct )
```

```
        BEGIN   CR ." Please enter your guess."
                GetNumber
                2DUP =                  ( equal? )
                IF      2DROP           ( discard both numbers )
                        CR ." Correct!!!"
                        EXIT
                THEN
                OVER <
                IF      CR ." Too low."
                ELSE    CR ." Too high!"
                THEN    CR
        0 UNTIL                         ( always repeat )
        ;

: Greet ( -- )
        CR CR CR ." GUESS A NUMBER"
        CR ." This is a number guessing game.  I'll think"
        CR ." of a number between 0 and any limit you want."
        CR ." (It should be smaller than 32000.)"
        CR ." Then you have to guess what it is."
        ;

: GUESS ( -- , the game )
        Greet
        BEGIN   InitialNumber                   ( set initial number)
                Check                           ( let player guess )
                CR CR ." Do you want to play again? (Y/N) "
                KEY                             ( get one key )
                32 OR 110 =                     ( exit if it is N or n )
        UNTIL
        CR CR ." Thank you.  Have a good day."  ( sign off )
        CR
        ;


( Type 'GUESS' will initialize the game and the computer will entertain
a user for a while.  Note the use of the indefinite loop structure:

        BEGIN <repeat-clause> [ f ] UNTIL

You can jump out of the infinite loop by the instruction EXIT, which
skips all the instructions in a Forth definition up to ';', which
terminates this definition and continues to the next definition. )

\ end
```