8-PUZZLE  See website link.

**Homework 25.1(8)**  l5_puzzle.f95. Write a program for
playing the 15-puzzle.
email: dale@math.hawaii.edu subject line: 190 h25.1(8)

**Classwork 25.1(4)**  8_puzzle.f95. Write a program for
playing the 8-puzzle.

We represent the 8-puzzle board with a matrix \( b \) with 0
being the “hole”. An initial matrix might be

\[
\begin{pmatrix}
1 & 2 & 3 \\
4 & 5 & 7 \\
8 & 6 & 0
\end{pmatrix}
\]

and we slide blocks around to get the goal

\[
\begin{pmatrix}
1 & 2 & 3 \\
4 & 5 & 6 \\
7 & 8 & 0
\end{pmatrix}
\]

For any given board, each numbered block \( n \) on the board
will have a position. Let \([i_0,j_0]\) be the position of the hole,
\(i.e., 0\). For this initial board \([i_0,j_0]=[3,3]\).

Moving a block into the hole is equivalent to swapping the
positions of the block and the hole and this is equivalent to
moving the hole into the block’s position. The position
\([inxt,jnxt]\) of the block will be the next position of the
hole. The user uses arrow keys to move the hole around.

Your homework deals with a \(4 \times 4\) matrix of 15 sliding
blocks with the goal being

\[
\begin{pmatrix}
1 & 2 & 3 & 4 \\
5 & 6 & 7 & 8 \\
9 & 10 & 11 & 12 \\
13 & 14 & 15
\end{pmatrix}
\]

You may pick any
board other than the goal board for your initial board.
Initialize \([i_0,j_0]\) to the initial board’s hole position.

```fortran
!c25_1_48_puzzle.f95
program eight_puzzle
integer::b(3,3),goal(3,3)
integer::m=0,n,i0=3,j0=3,inxt,jnxt
b(1,:)=(/1,2,3/); goal(1,:)=[1,2,3]
b(2,:)=(/4,5,7/); goal(2,:)=[4,5,6]
b(3,:)=(/8,6,0/); goal(3,:)=
5 format(3(i2))
print 5,(b(i,:),i=1,3)
print*,"Enter numpad arrow key or 5. 0 quits."
do
read *,n
if(n==0)exit
m=m+1; inxt=i0; jnxt=j0
select case(n)
   case(5,2); inxt=min(i0+1,3)
   ... three more lines needed here
endselect
call swap( _____ , _____ )
i0=inxt; j0=_____
print 5,(b(i,:),i=1,3)
if(all(b==goal))then;
   print*,'Solved in',m,' moves';exit;
endif
endo
dendprogram

subroutine swap(i,j);
k=i; i=j; j=k;
```

Tic Tac Toe

As with the 8-puzzle, we represent the board with a 3×3 matrix. The user plays a position by entering the position on the keyboard numpad. Positions already played are marked X or O. Here is a typical board:

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>O</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Four blanks are cycle, exit, Game is a draw.

Classwork 25.2(4) tictactoe.f95

CLASSWORK 25.1(4) 8_puzzle 25.2(4) tictactoe

e-mail: dale@math.hawaii.edu  subject line: 190 c25(8)

Homework 25.1(8) 15_puzzle
The Tower of Hanoi problem has three pegs A,B,C. Disks of various sizes sit on the pegs. Rule: a larger disk may not sit on top of a smaller disk. In one move you can move the top disk on one peg to another peg provided the rule is not violated. Given some number \( n \) of disks on peg A, and none on B or C, move them all to peg C.

First build a mathematical representation of the problem. Number pegs A,B,C as peg 1,2,3. Number the disks 1,2,3, ..., \( n \) with the larger numbers representing the larger disks. Represent the configuration of the pegs with a matrix and vector integer ::\( \text{peg}(3,100) \) so that row \( i \) of the matrix \( \text{peg} \) lists of disks on peg \( i \) listed from bottom up.

If \( n=4 \), the initial configuration has 4 disks on A.

Rotating the picture on its side suggests representing the configuration with a matrix of three rows with the the disks on a peg listed in the peg’s row.

It is represented by
\[
\text{peg} = \begin{bmatrix}
4 & 3 & 2 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0
\end{bmatrix}
\]

The configuration below

is represented by the matrix
\[
\text{peg} = \begin{bmatrix}
4 & 3 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
2 & 1 & 0 & 0 & 0
\end{bmatrix}
\]

\( \text{peg}(1,:) \) is peg A, \( \text{peg}(2,:) \) is peg B, \( \text{peg}(3,:) \) is C.
**Classwork 26.1(5)** towerhanoi.f95

```fortran
program tower_of_hanoi
integer:: peg(3,100)=0
integer:: ndisks, frompeg, topeg
peg(1,1:4)=(/4,3,2,1/)
!peg(i,:) = disks on peg i. todisk(i) = number of top disk on peg i
    call print_peg_configuration
    call peg_moves(4,1,3)
!peg_moves( 4, 1, 3) means move 4 disks from peg 1 to peg 3
contains
    recursive subroutine peg_moves(ndisks,frompeg,topeg)
    integer:: disk, ndisks, frompeg, topeg, otherpeg
    if(ndisks==0) return
    otherpeg=6-frompeg-topeg !frompeg+otherpeg+topeg=6
    ... replace this with two lines
    call print_peg_configuration
    call peg_moves(ndisks-1,otherpeg,topeg)
    end subroutine
    subroutine move_disk(frompeg,topeg)
    integer:: frompeg, topeg, disk
    disk=peg(frompeg,todisk(frompeg))
    peg(frompeg,todisk(frompeg))= ______________
    peg(topeg,todisk(topeg)+1)= ______________
    end subroutine
    integer function todisk(i)
    todisk=count(peg(i,:)>0)
    endfunction
    subroutine print_peg_configuration
    3 format(a," ",10i2)
    print 3,' A', peg(1,1:todisk(1))
    print 3,' B', peg(2,1:todisk(2))
    print 3,' C', peg(3,1:todisk(3))
    print 3,'--------'
end subroutine
end program
```

**Homework 26.1(3)** towerhanoi2.f95. Write a program for solving the Tower of Hanoi puzzle where the initial peg has 5 disks instead of 4 as in the classwork problem. See website link.