

POINTS TO PONDER

4.1 The function

```

INTEGER FUNCTION TEST(BOARD,I,J)
INTEGER BOARD(8,8)
TEST = -1
IF (I.LT.1 .OR. I.GT.8 .OR. J.LT.1 .OR. J.GT.8) RETURN
IF (BOARD(I,J).GE.0 .AND. BOARD(I,J).LE.4) TEST = BOARD(I,J)
RETURN
END

```

returns `-1` if `BOARD(I,J)` is undefined or illegal; otherwise it returns `BOARD(I,J)`. Rewrite subroutine `STORE` of the checker-playing program to use `TEST`. Add code to `SEARCH` to include valid single jumping moves. At most eight statements should have to be added. (Don't forget to add `TEST` to the `INTEGER` statement and increase the sizes of `ROW` and `COL`.) What would be involved in adding jumps to the original version?

4.2 What simplifications can be made in the checker-playing subroutines if we use a 10 by 10 checkerboard, where the border squares contain negative values?

4.3 What happens to the original maze program if the top border looks like

```

...010...
...000...
.....

```

What if the top left corner looks like

```

110.....
000.....
.....

```

The `POINT` array can handle up to 60 loops. Can you define a maze that has more than 60 loops? Remember that

```

.....
1111111111111111
010101010101010
000000000000000

```

looks like seven loops when searched from left to right. Does our version handle these cases correctly?

4.4 Rewrite the maze program without using recursion, using the recursive version as a model. How much bigger and more complicated does it get? Can you do better with a different approach?