

```

MOUSE: PROCEDURE OPTIONS (MAIN);
  /* MOUSE IN A MAZE */
  /* SEARCHES PERIPHERY OF M X N MATRIX FOR AN ENTRY POINT */
  /* FIRST PATH WITH EXIT DIFFERENT FROM ENTRANCE IS ACCEPTED */

  DECLARE (YES INITIAL('1'B), NO INITIAL('0'B)) BIT(1);
  DECLARE MAZE(50,50) BIT(1);
  DECLARE WALL BIT(1) INITIAL('0'B);
  DECLARE STATE(50,50) CHARACTER(1);
  DECLARE USED CHARACTER(1) INITIAL('U');
  DECLARE FREE CHARACTER(1) INITIAL('F');
  DECLARE (PATHPTR, IPATH(2000), JPATH(2000)) FIXED BINARY;
  DECLARE (M, N) FIXED BINARY;

  DO WHILE (READMAZE() = YES);
    IF FINDPATH() = YES THEN
      CALL PRINTPATH;
    ELSE
      PUT SKIP(2) LIST ('NO PATH');
    END;
  END;

READMAZE: PROCEDURE RETURNS (BIT(1));
  ON ENDFILE(SYSIN)
    GOTO EOF;
  GET LIST (M, N);
  IF M < 2 | M > 50 | N < 2 | N > 50 THEN DO;
    PUT SKIP LIST (M, N, 'BAD DIMENSIONS');
    RETURN(NO);
  END;
  GET EDIT ((MAZE(I,J) DO J = 1 TO N) DO I = 1 TO M))
    (COLUMN(1), (N)B(1));
  PUT PAGE EDIT ((MAZE(I,J) DO J = 1 TO N) DO I = 1 TO M))
    (COLUMN(1), (N)B(1));
  RETURN(YES);
EOF:
  RETURN(NO);
END READMAZE;

FINDPATH: PROCEDURE RETURNS (BIT(1));
  STATE(*, *) = FREE;
  PATHPTR = 0;
  DO I = 2 TO M-1;
    IF TRY(I, 1, I, 2) THEN /* LEFT SIDE */
      RETURN(YES);
    IF TRY(I, N, I, N-1) THEN /* RIGHT SIDE */
      RETURN(YES);
  END;
  DO J = 2 TO N-1;
    IF TRY(1, J, 2, J) THEN /* TOP */
      RETURN(YES);
    IF TRY(M, J, M-1, J) THEN /* BOTTOM */
      RETURN(YES);
  END;
  RETURN(NO);
END FINDPATH;

```