


```

(* the rest of random here *)
begin (* main program *)
  seed := 474747.0; (* or any value from 1.0 to 2^31-1 *)
  (* now make use of random *)
end.

```

- (a) Write a segment of code which will use this random number generator to print 10 random real numbers between 0.0 and 1.0.
- (b) Write a code segment that will use `random` to simulate flipping a coin. Your segment should print out “HEADS” half the time and “TAILS” half the time.
- (c) Write a segment of code that will use the random number generator to produce *pairs* of numbers (x,y) , where both x and y are in the range from -0.5 to 0.5. As they are generated, count the number of pairs satisfying $x^2 + y^2 \leq 0.25$. Suppose this count is stored in a variable `COUNT`. Your segment should continue for N pairs, where N is 1000, say. After N pairs altogether, the segment should finally print the ratio `COUNT/N`. (This ratio is a monte-carlo approximation to $\pi/4$.)

- (35) 3. Suppose we have an array type declared as follows:

```

const N = 4;
type Atype = array[1..N] of integer;

```

- (a) Show how to declare 3 global arrays `A`, `B`, and `C`, each of N integers.
- (b) Write a procedure `READ_ARR` with one parameter of type `Atype` that will read N numbers from the terminal and insert them into its parameter. (Don't worry about end-of-file.)
- (c) Write a function `SUM` with one parameter of type `Atype` that will add up the N integers in its parameter and return the sum as the function value. Thus if `A` holds values 2, 3, 1, and 4, then the reference `SUM(A)` would return 10 as its value.
- (d) Consider the procedure `PROD` below

```

PROCEDURE PROD (X, Y: Atype; VAR Z: Atype);
VAR I: integer;
BEGIN
  FOR I := 1 TO N DO
    Z[I] := X[I]*Y[I]
  END;

```

If `A` has values 2, 3, 1, and 4, and `B` has values 1, 4, 0, and 2, then after the call `PROD(A, B, C)`, what will the values in `C` be? (Show how you get the answer.)

- (e) Write a function `INNER_PROD` with two input parameters of type `Atype`. `INNER_PROD` should return as its value the sum of the products of corresponding array positions. Thus with `A` and `B` with values as in (d) above, `INNER_PROD` should calculate and return the value

$$2*1 + 3*4 + 1*0 + 4*2 = 22.$$

For full credit, `INNER_PROD` should use both `PROD` and `SUM` above.