RANDOM NUMBER GENERATION:

Suppose that we have a means of selecting an integer r at random from the set of integers 1,2,...,N. Simple mechanisms exist for performing this selection for certain small values of N. For N=2 we can toss a coin, for N=6 we can roll a die, and for N=36 we can use a roulette wheel. A sequence

r_1, r_2, ...

of such integers is called a *random sequence*. An algorithm which generates a random, or apparently random, sequence is called a *random number generator*. Several of the examples in our textbook (See pages A199-A200 and 305-307) require a random number generator.

The method most commonly used to generate random numbers is the *linear congruential method*. Each number in the sequence, r_k, is calculated from its predecessor, r_{k-1}, using this formula:

r_k = (multiplier * r_{k-1} + increment) MOD modulus

(Note: Recall that the MOD function essentially gives the remainder after a division of the arguments.) The numbers generated by this formula repeatedly are not truly random in the sense that tosses of a coin or throws of a die are random, because we can always predict the value of r_k given the value of r_o. The sequence generated by this formula is therefore more correctly called a pseudo-random sequence, and its members are called pseudo-random numbers. There are many applications of random numbers in computer science and engineering, and for most of these pseudo-random numbers are just as good provided that some elementary precautions are taken.

Most computer installations have a standard pseudo-random number generator of the linear congruential type, for which the values of multiplier, increment, and modulus have been carefully chosen. Since our assembly language development environment is rather limited in its capabilities (and we need some extra credit problems), consider the following Pascal algorithm which generates 65536 random numbers before repeating itself, and is adequate for most applications. We use:

modulus = 2^{16} = 65536
multiplier = 25173
increment = 13849

and so we have

r_k = (25173 * r_{k-1} + 13849) MOD 65536

This calculation will not cause overflow on a computer for which maxint ≥ 2^{31}-1

The pseudo random number generator is written as a Pascal function.

```
FUNCTION random (VAR seed : integer) : integer;
        CONST
                multiplier = 25173;
                increment = 13849;
                modulus = 65536;
        BEGIN
                random := seed;
                seed := (multiplier * seed + increment) MOD modulus
        END;
```

Note that this function contravenes the rule that a function should not change the value of any of its parameters. Since we want random to return a different value each time we call it, this is unfortunately unavoidable. To save space, when we use random in examples, we will use literals rather than constants for multiplier, increment, and modulus. The function generates a permutation of the integers 0,1,2, ... , 65535

and then repeats itself. The first number generated is the initial value of seed.

For most applications these numbers should be scaled in some way. For example, if the program is calculating throws of a die, we could write

```
        VAR
                throw, seed : integer;
        ....
                throw := random (seed) MOD 6 + 1
```

We frequently require a real random value between 0 and 1. For this purpose we can use a modified version of random:

```
FUNCTION random (VAR seed : integer) : real;
        BEGIN
                random := seed / 65535;
                seed := (25173 * seed + 13849) MOD 65536
        END;
```

Assignment:
Write an assembly language program that generates a pseudo random number sequence using the linear congruential method. This method should be functionally identical to the routines SEED and RAND described on page 305 of Ford & Topp, Second Edition.

Reference:
Peter Grogono, Programming in Pascal, Section 4.5