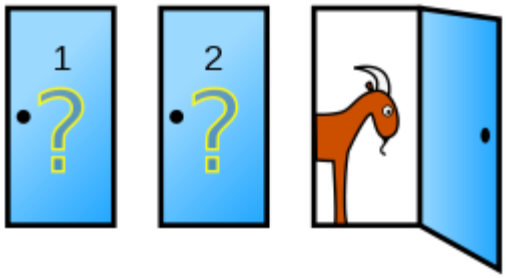
Task 2. Monty Hall Problem

Available marks: 16

Monty Hall was the original host of the game show *Let's Make a Deal*. One of his (apocryphal) deals, popularised by columnist Marilyn vos Savant goes like this:

> *A car and two goats are hidden behind three closed doors numbered 1, 2 and 3. Hoping to win the car, you choose one door at random, say door 2. Monty then opens one of the other doors, say door 3, to reveal a goat, then says: "Do you want to keep your original choice or switch your choice to the other closed door?"*
>
> *Should you stay, should you switch, or doesn't it matter?*



You will solve this problem experimentally, using the pseudo-random number generator specified below. You could use the one that comes with your language system, but this way the results are predictable, and you can easily adapt it later if you think we've biased the results (we haven't). The same one is used for Task 4.

Random number generator

The generator you must use for this task and also for Task 4 is a multiplicative linear congruential generator, defined as follows.

- S_i is the i-th integer seed (maximum 24 bits),
- u_i is a uniformly-distributed pseudo-random value between 0 and 1 (excluding 1), and
- $r_{i,N}$ is a uniformly-distributed pseudo-random value between 0 and N, excluding N.

$S_{i+1} = (a\ S_i) \bmod m$
where $S_0 = 1$, $a = 6423135$, $m = 16777213$.

$u_i = S_i / m$

$r_{i,N} = \text{floor}(u_i * N)$, where $\text{floor}(x)$ is the largest integer less than or equal to x.

You only need to remember the current seed and the parameters a and m, not the entire sequence.

The three functions you must code are

```
myrand() - advances i and returns u_i
myrandint(N) - returns r_{i,N}, calling myrand() once
myrandperm(N) - generates a pseudo-random permution of N items
  using this algorithm (assumes zero-based indexing notation [ ]):
  for pos = 0 to N-2
      swap perm[pos] with perm[pos + myrandint(N-pos)]
```

Note 1: the very first call on `myrand` uses S_1, not S_0.

Note 2: `myrandperm` calls `myrandint` exactly *N*-1 times.

Experimental method

The participants in the game are the host **Monty**, who knows what's behind the doors, and three personas of the player, based on characters from *Alice's Adventures in Wonderland*: **Dormouse**, who's mostly asleep and never changes his mind, **Hatter**, who always wants to switch (remember the tea party, where Hatter and his mates keep changing places to avoid having to wash up), and **Alice**, who makes a random choice of switching or not ("Which path shall I follow?", asks Alice at the crossroads. The Cheshire Cat answers, "That depends where you want to go. If you do not know where you want to go, it doesn't matter which path you take.")

Your program must follow this algorithm precisely, using appropriate calls on the implemented myrandperm() and myrandint() functions.

```
Repeat 10 times:
    Randomly permute the sequence { car, goat, goat }
    Display the permutation

Reset the random number generator
Repeat 1500 times:
    Randomly permute the sequence { car, goat, goat }
    (keep count of how many times the car appears behind each door)
    Randomly choose a door for Dormouse
    Depending on Dormouse's choice, open one of the doors
        (either the other goat if Dormouse has chosen a goat,
          or a random choice between the goats otherwise)
    Assign the remaining door to Hatter
    Randomly choose between Dormouse's and Hatter's door and assign that to Alice
    Give Dormouse 1 point if he has chosen the car or 1 to Hatter otherwise
    Give Alice 1 point if she has chosen the car
```

At the end of the experiment, report the number of times the car appears behind each door as a distribution check, then Dormouse's, Hatter's and Alice's total points.

If your generator is working, the first 10 permutations will match the list below. Each iteration of the main loop will result in either 4 or 5 calls on myrand().

```
goat car goat
car goat goat
car goat goat
goat car goat
goat goat car
goat car goat
goat goat car
car goat goat
goat car goat
goat goat car
```

Assessment

There is no test data for this task. When you believe you have completed it show the judges your output and write below (so other teams don't hear) whether the player should always stay, always switch or toss a coin.

_____

Marking Scheme

3 marks for the correct 10 permutations,
10 marks for the players' simulation results (a close match to ours is acceptable)
2 marks for the car distribution results (a close match to ours is acceptable)
1 mark for your written answer, one chance only at this.

References:

Marilyn vos Savant, Game Show Problem
Monty Hall problem (Wikipedia). Image by user:Cepheus.
L'Ecuyer, P (1999). "Tables of Linear Congruential Generators of Different Sizes and Good Lattice Structure", *Mathematics of Computation* Vol **68**, No. 225, pp249-260.