Problem S1J3

Slot Machines

Martha takes a jar of quarters to the casino with the intention of becoming rich. She plays three machines in turn. Unknown to her, the machines are entirely predictable. Each play costs one quarter. The first machine pays 30 quarters every 35th time it is played; the second machine pays 60 quarters every 100th time it is played; the third pays 9 quarters every 10th time it is played.

Your program should take as input the number of quarters in Martha's jar (there will be at least one and fewer than 1000), and the number of times each machine has been played since it last paid.

Your program should output the number of times Martha plays until she goes broke.

Input/output is not from/to files for this question. Keyboard input and screen output is expected.

Sample Session. *User input is in italics*

```
How many quarters does Martha have in the jar?
48

How many times has the first machine been played since paying out?
3

How many times has the second machine been played since paying out?
10

How many times has the third machine been played since paying out?
4

Martha plays 66 times before going broke.
```

QOJ adaptation notes

Input: read the initial number of quarters q (1 <= q < 1000), followed by the numbers of plays since the last payout for machines 1, 2 and 3. These counters are respectively in 0..34, 0..99 and 0..9.

Play machine 1 first, then 2, then 3 repeatedly. Pay a quarter and apply any payout from that play before deciding whether Martha is broke. Output: Martha plays N times before going broke. Replace N with the total number of plays.

Test data and scoring: the original official hidden inputs for this problem could not be recovered. QOJ therefore supplies newly constructed tests; these are NOT the original CCC hidden tests. The tests cover boundary cases and varied valid inputs, and have been checked with an independent oracle.

The QOJ reconstructed score is 15 points: five all-or-nothing batches worth 3 points each. These batch assignments are QOJ reconstructions, not claims about the unavailable historical test selection.

Coverage includes near-payout counters, immediate bankruptcy, simultaneous low balances and payouts, maximum initial balance, and deterministic random states. Oracle: jump between scheduled payout events, independently of the reference solution that simulates individual plays.