National University of Singapore
School of Computing
CS1010X: Programming Methodology
Semester II, 2019/2020

Contest 7.2
More Lazy Susan

Release date: 29 February 2020
Due: 19 April 2020, 23:59

Required Files

- contest07.2-template.py
- more_lazy_susan.py

Background

Hours have passed. You decide to head to the kitchen to grab supper. Only... you can't seem to find the door. And it seems to be getting a little hazy. You have a bad feeling about this. As if confirming your thoughts, a familiar genie manifests through the smoky haze and presents you with... more Lazy Susan.

Mission Brief

The genie, it appears, was impressed with your earlier performance on the train, and is offering you an invitation to his contest.

In this contest, you will compete to create the best solver for the *Lazy Susan Problem* with *10, 13, and 15 coins*.

Entries will be judged based on how many steps the solver takes to flip all coins to the same orientation. Your solver will be run for 3000 rounds; 1000 rounds each for 10, 13, and 15 coins. The final score will be computed as the sum of the scores obtained across all 3000 rounds.

You are only allowed one entry. There will be a move limit of 4000 moves per round, after which the round will be considered unsolved and you will obtain 0 score for that round.

The template file `contest07.2-template.py` has been provided for you. In addition, another file `more_lazy_susan.py` has also been provided. It contains functions that will run and score your solver.

After you have completed your solutions in the template file, copy only your function to coursemology. Do not `import more_lazy_susan`.

Note: If you require the use of any functions in `more_lazy_susan`, copy the functions directly over to coursemology.

Task

Write a higher-order function `create_solver(coins)` that takes in the number of coins and returns a solver function. This function, `solver(move_id)`, takes in the current move number, and returns a boolean array representing the move to be made. For each `True` value in the array, the corresponding coin at the same index will be flipped. This boolean array must be of length `coins`.

For each round, the `move_id` will begin at 0 and increment with each move. You can assume that no two rounds will be played concurrently and that a `move_id` of 0 always indicates the start of a new round.

Testing Your Solver

Uncomment the below line to grade your solver according to the competition criteria.

```
get_contest_score(create_solver, True)
```