Spencer Ward **September 14, 2016** **Homework 1**

Problem 1:

```
b = " "

a = "*"


print b*2 + a*2 + b*12 + a + b*5 + a + b*3 + a*3 + b*5 + a*2 + b*5 + a*3 + b*3

print b + a + b*2 + a + b*11 + a + b*5 + a + b*2 + a + b*3 + a + b*3 + a + b*2 + a
+ b*4 + a + b*2 + a + b*2

print b + a + b*14 + a + b*5 + a + b*2 + a + b*3 + a + b*3 + a + b*2 + a + b*4 + a
+ b*3 + a + b

print b*2 + a*2 + b*12 + a + b*5 + a + b*2 + a + b*3 + a + b*3 + a + b*2 + a + b*4
+ a + b*3 + a + b

print b*4 + a + b*12 + a + b + a + b + a + b*3 + a*5 + b*3 + a*3 + b*5 + a + b*3 +
a + b

print b + a + b*2 + a + b*5 + a*2 + b*5 + a+ b + a + b + a + b*3 + a + b*3 + a +
b*3 + a + b + a*2 + b*4 + a + b*2 + a + b*2

print b*2 + a*2 + b*6 + a*2 + b*6 + a + b + a + b*4 + a + b*3 + a + b*3 + a + b*2
+ a + b*4 + a*3 + b*3
```

```
RESTART: D:/Users/SWK/Desktop/Google Drive/Documents/2A College Work Freshman/COS
125/hw1-1.py

  **            *     *   ***     **     ***

 *  *           *     *  *   *   *  *    *  *

 *              *     *  *   *   *  *    *   *

  **            *     *  *   *   *  *    *   *

    *            * * *   *****   ***     *   *

 *  *     **     * * *   *   *   * **    *  *

  **      **      * *    *   *   *  *    ***

>>>
```

Problem 2:

Wins with swapping: 20

Wins without swapping: 8

The win rate with swapping is higher because there is a 1 in 3 chance that the prize is behind the first curtain you choose, and a 2 in 3 chance that it is behind one of the two you did not choose. After the empty curtain is removed, there is still a 2 in 3 chance that the prize is behind the one remaining curtain.

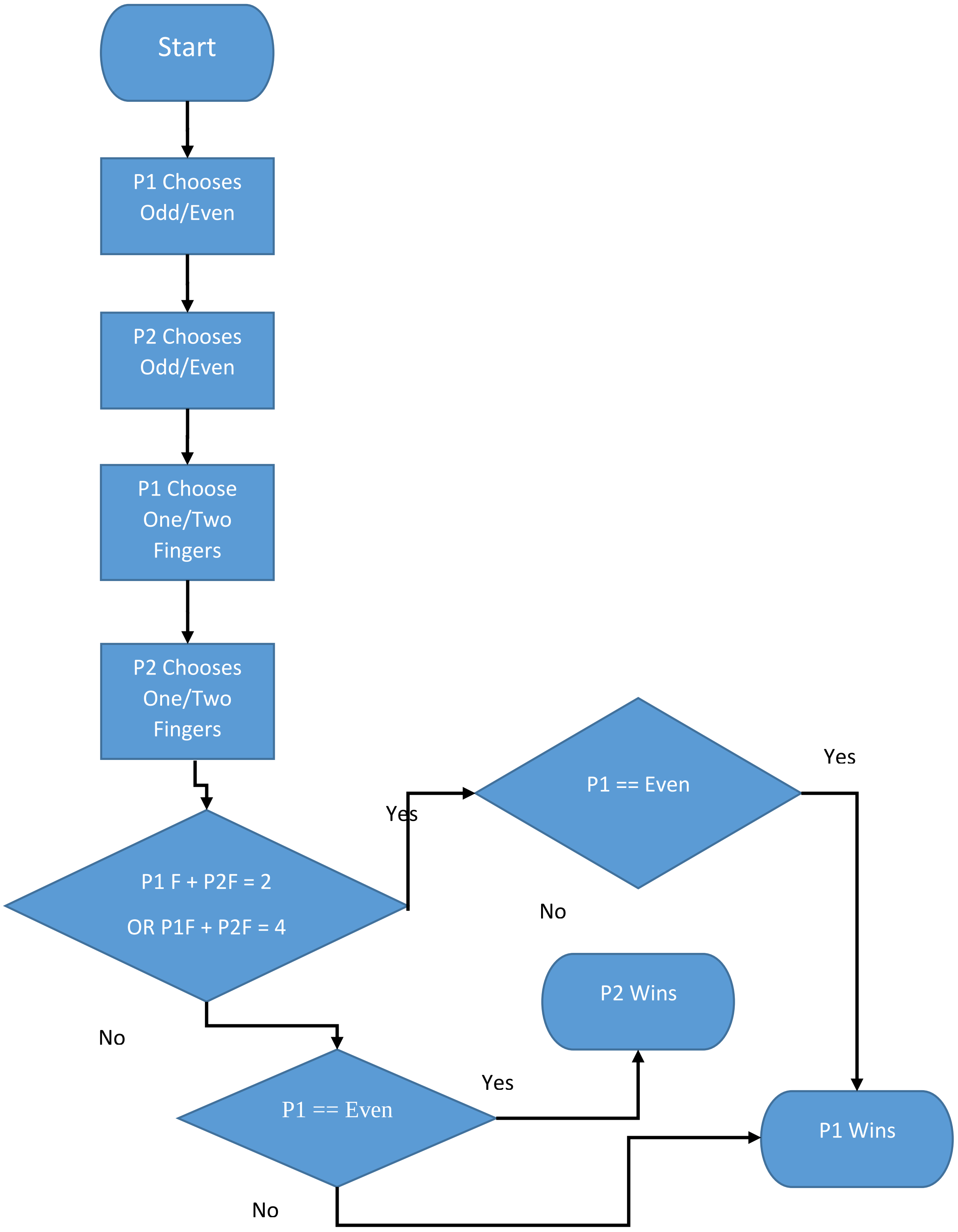
Another way to explain the win rate is that there is a 1 in 3 chance that the prize is behind the curtain you chose, but a 2 in 3 chance that it is behind one of the other two. If the prize is indeed behind one of the others, the game is forced to remove the empty one, meaning that there is a 2 in 3 chance that the game reveals the curtain with the prize.

Problem 3:

```
import random

while True:
    wins = 0

    ynr = raw_input('Do you want to swap curtains' +
        '? (y/n/r) ')

    for i in range(1000):
        choices = ['1', '2', '3']
        car = random.choice(choices)
        human = random.choice(choices)

        choices.remove(car)
        if human in choices:
            choices.remove(human)

        empty = random.choice(choices)

        choices = ['1', '2', '3']
        choices.remove(empty)
        choices.remove(human)

        if ynr == 'r':
            yn = random.choice(['y', 'n'])
        else:
            yn = ynr

        if yn == 'y' :
            human = choices[0]
        if human == car:
            wins += 1
    print wins, "/ 1000 Wins"
```

```
RESTART: D:/Users/SWK/Desktop/Google Drive/Documents/2A College Work Freshman/COS
125/hw1-3.py
Do you want to swap curtains? (y/n/r) y
668 / 1000 Wins
Do you want to swap curtains? (y/n/r) n
335 / 1000 Wins
Do you want to swap curtains? (y/n/r) r
504 / 1000 Wins
```

Problem 4:


Start
P1 Chooses Odd/Even
P2 Chooses Odd/Even
P1 Choose One/Two Fingers
P2 Chooses One/Two Fingers
P1 F + P2F = 2
OR P1F + P2F = 4
Yes
P1 == Even
Yes
No
No
P2 Wins
P1 == Even
Yes
P1 Wins
No

Spencer Ward **September 14, 2016** **Homework 1**

Problem 5:

A Python Enhancement Proposal (PEP) is a formal design document describing and providing rationale for the addition of a new feature to Python or one of its underlying components. These documents are intended to give the Python community insight on the design decisions that go into the language they use, as well as give the developers an opportunity to collect feedback to adjust, or if necessary, cancel the feature.

Problem 6:

Python value for `x`: `0.20000000000000018`

Python produces results that appear inaccurate when performing decimal floating-point arithmetic because computers must store decimal fractions as binary conversions. This means that many ordinary fractions, in this case 0.2, need to be converted from the decimal 2/10 to a binary summation of a/2 + b/4 + b/8, and so on. For some fractions, a direct conversion does exist, but in most cases the computer needs to approximate to values very close but not equal to the original decimal values.

PROBLEM N:

`Python.org: PEP 1 -- PEP Purpose and Guidelines`

`https://www.python.org/dev/peps/pep-0001/`

`Python.org: Floating Point Arithmetic: Issues and Limitations`

`https://docs.python.org/2/tutorial/floatingpoint.html`