Lab 2 Report Template

1. In coffee can, there are n black beans and m white beans, each time two beans will be taken out: if they are with same color, throw away both, then put in one black bean; if they are with different color, throw away the black bean, put back the white bean.

Now, a) will this game stop?

b) for what kind of combination (n, m), the last one left must be a white bean?

A: This game will stop since whatever the situation of each turn of this game, while the number of all of the beans bigger than one, the number will reduce one after a turn of game. So this game will stop.

B: For each time two beans will be taken out:

If there are two white beans, the number of white beans reduce 2 and the number of black beans increase 1.

If there are two black beans, the number of black beans reduce 1.

If there are two different beans, the number of black beans reduce 1.

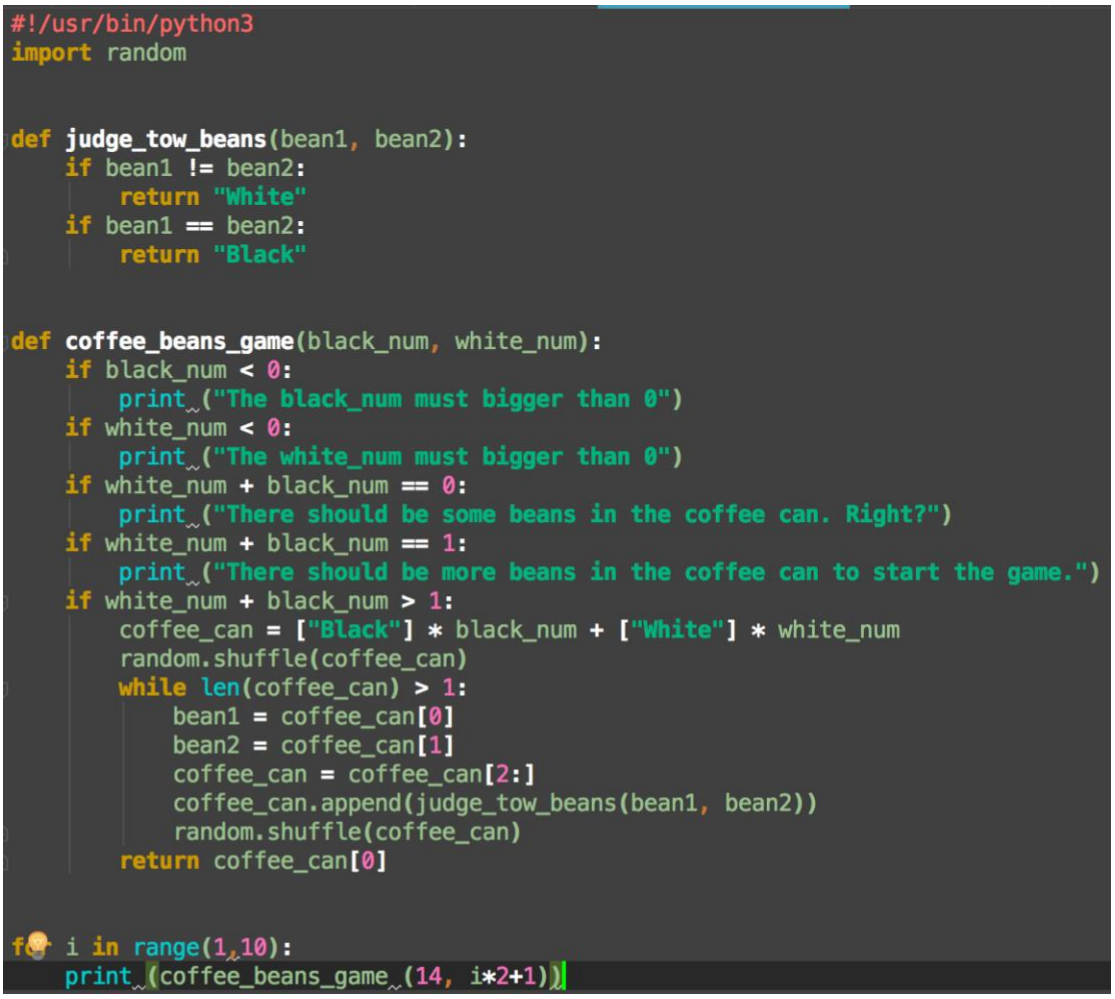
And we know that the game will stop only if a single bean leave, so if the number of while beans, the m, is an odd number, the game will stop and the last one left must be a white bean

So, for the combination (n, m):

n >= 0 and m = 2k+1 (k is a natural number)

I used the Python to run this game and I test the result of odd number form in range 3 to 19.

And the program will be showed below:



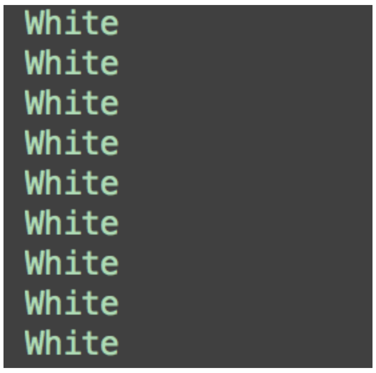
```python
#!/usr/bin/python3
import random


def judge_tow_beans(bean1, bean2):
    if bean1 != bean2:
        return "White"
    if bean1 == bean2:
        return "Black"


def coffee_beans_game(black_num, white_num):
    if black_num < 0:
        print ("The black_num must bigger than 0")
    if white_num < 0:
        print ("The white_num must bigger than 0")
    if white_num + black_num == 0:
        print ("There should be some beans in the coffee can. Right?")
    if white_num + black_num == 1:
        print ("There should be more beans in the coffee can to start the game.")
    if white_num + black_num > 1:
        coffee_can = ["Black"] * black_num + ["White"] * white_num
        random.shuffle(coffee_can)
        while len(coffee_can) > 1:
            bean1 = coffee_can[0]
            bean2 = coffee_can[1]
            coffee_can = coffee_can[2:]
            coffee_can.append(judge_tow_beans(bean1, bean2))
            random.shuffle(coffee_can)
        return coffee_can[0]


for i in range(1,10):
    print (coffee_beans_game (14, i*2+1))
```

The output:



```
White
White
White
White
White
White
White
White
White
```

2. Given n integers, in worst case, can they be sorted in less than O(n) time? Why?

They cannot be sorted in O(n) time, since in the worst case the step that find the max need n-1 times of calculation, the time cost in every turn is not more than O(n), and number of the turns relevant to the n, so in the worst case the time complexity is O(n*f(n)), where f(n) is a function of n. So the process of sort cannot be finished in O(n) time.

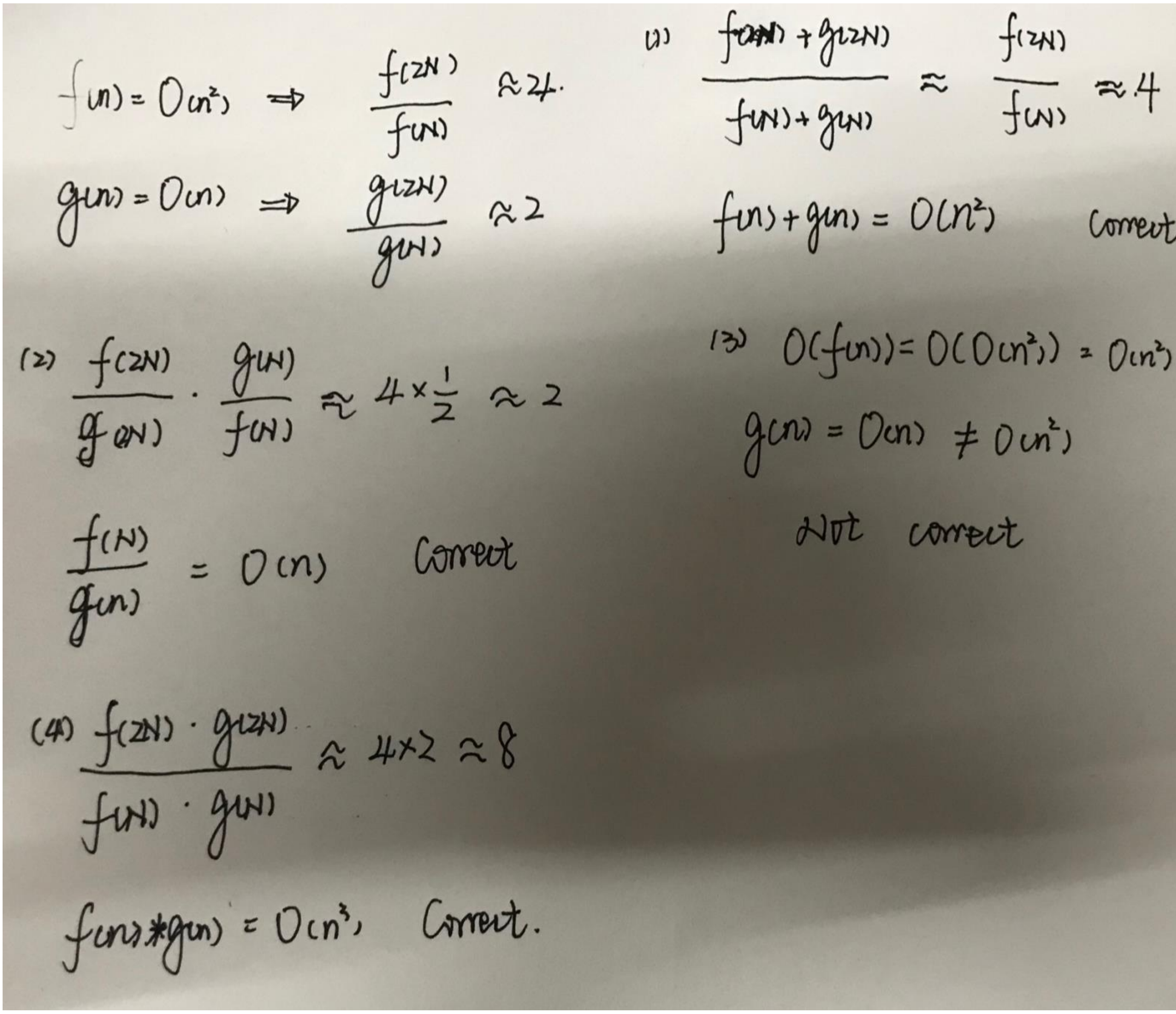
3. If f(n) = O(n^2) and g(n) = O(n), show whether the following is correct or not:

1) f(n) + g(n) = O(n^2) Correct

2) f(n)/g(n) = O(n) Correct

3) g(n) = O(f(n)) Not Correct

4) f(n) * g(n) = O(n^3) Correct



$f(n) = O(n^2) \Rightarrow \frac{f(2N)}{f(N)} \approx 4.$

$g(n) = O(n) \Rightarrow \frac{g(2N)}{g(N)} \approx 2$

(1) $\frac{f(2N) + g(2N)}{f(N) + g(N)} \approx \frac{f(2N)}{f(N)} \approx 4$

$f(n) + g(n) = O(n^2)$ Correct

(2) $\frac{f(2N)}{g(2N)} \cdot \frac{g(N)}{f(N)} \approx 4 \times \frac{1}{2} \approx 2$

$\frac{f(N)}{g(n)} = O(n)$ Correct

(3) $O(f(n)) = O(O(n^2)) = O(n^2)$

$g(n) = O(n) \neq O(n^2)$

Not correct

(4) $\frac{f(2N) \cdot g(2N)}{f(N) \cdot g(N)} \approx 4 \times 2 \approx 8$

$f(n) * g(n) = O(n^3)$ Correct.

4. Performance measurement of 4 algorithms with the same function and different time complexities.

Algorithm		**1**	**2**	**3**	**4**
Time(seconds)		**O(*N*^{3})**	**O(*N*^{2})**	**O(*N*log*N*)**	**O(*N*)**
Input Size	**N=10**	**0.0000813**	**0.0000322**	**0.0000498**	**0.0000257**
	N=100	**0.0350730**	**0.0009999**	**0.0004320**	**0.0000441**
	N=1000	**33.488880**	**0.0960209**	**0.0055439**	**0.0003321**
	N=10000	**NA**	**9.2699549**	**0.0653030**	**0.0040559**
	N=100000	**NA**	**NA**	**0.6084790229**	**0.0374195**