Homework #3

Analysis of Algorithms

Illya Starikov

Due Date: September 19^{th}, 2017

Question #1

```python
def fn(n):
    if n > 100:
        return n - 10
    return ((n - 101) % (c - 10)) + 91
```

Question #2

The following function tests random values up to in $\{-1\,000\,000, \ldots, 1\,000\,000\}$. `is_gcd_fast` is a faster alternative because $\gcd \in \mathcal{O}\left(\log_{\frac{a}{a \mod b}}(a \times b)\right)$ while $\gcd_{\text{fast}} \in \mathcal{O}(c)$.

```python
from fractions import gcd
import random


def is_gcd(gcd_value, a, b):
    return gcd_value == gcd(a, b)


def is_gcd_fast(gcd_value, a, b):
    return a % gcd_value == b % gcd_value == 0

for i in range(1000):
    a = random.randint(-1000000, 1000000)
    b = random.randint(-1000000, 1000000)
    g = gcd(a, b) if random.choice([True, False]) else random.randint(-1000000, 1000000)

    print(is_gcd_fast(g, a, b) == is_gcd(g, a, b))
```

Question #3

3.1 All Sums

```python
# Warning, for all lists (with the exclusion of [1], [0] and []) this function does not stop
def generate_all_sums(l):
    for element in l:
        if element == 0:
            return [0] + generate_all_sums(element[1:])
        else:
            return [element] + [generate_all_sums([element*element])]

    return l
```

3.2 $Sum(T) \notin \{$*all odds*$\}$

Take Sum to be the function that maps all elements to all of their multiples.

Theorem 1. *There is not set T such that $Sum(T) = \{\text{all odd integers}\}$.*

We prove so by proof of contradiction.

Proof. Suppose not. That is, suppose that is a set T such that $Sum(T) = \{\text{all odd integers}\}$. Set T, at a minimum, must have a single element (for the empty set $\varnothing$ cannot produce any integers). We take an arbitrary element from this set T and name it q. There are two forms for q $(\exists m \in \mathbb{Z})$,

$$q = \begin{cases} 2m & \Leftrightarrow q \text{ is even} \\ 2m+1 & \Leftrightarrow q \text{ is odd} \end{cases}$$

If q is even, there is already a contradiction. If q is odd, then we have the following contradiction. Supposing we sum q twice, we have the following form:

$$q = a(2m+1) + b(2m+1)$$

Because a and b are arbitrary, we take them to be 1.

$$\begin{aligned} q &= (2m+1) + (2m+1) \\ &= 4m + 2 \\ &= 2(2m+1) \end{aligned}$$

Because we know the sum of integers and multiplication of integers to be integers, $2(2m+1) \equiv 2n, \exists n \in \mathbb{Z}$, which has the form of an even integer.

This has lead us to a contradiction. Therefore, our hypothesis is false, concluding there is not set T such that $Sum(T) = \{\text{all odd integers}\}$.

□

3.3 Fast Algorithm For S

To get d from an arbitrary set S, we simply take the greatest common divisor from all the sets.

```
from functools import reduce

def gcd(numbers):
    def gcd_single(a, b):
        while b:
            a, b = b, a % b
        return a

    return reduce(gcd_single, numbers)
```

3.4 $d \in \{540051690381, 5404079462298, 3485942644184\}$

547.

Question #4

Theorem 2. $\forall i \in \mathbb{Z}^+$,

$$\sum_{i=1}^{n} (-1)^{i-1} i^3 = -\frac{1}{8}(-1)^n \left(4n^3 + 6n^2 - 1\right) - \frac{1}{8}$$

Proof. *Step #1* We wish to prove that for all natural numbers $f(n) = g(n)$, where f and g are as defined as follows:

$$f(x) = \sum_{i=1}^{n} (-1)^{i-1} i^3$$
$$g(x) = -\frac{1}{8}(-1)^n \left(4n^3 + 6n^2 - 1\right) - \frac{1}{8}$$

Let D be the set of natural numbers (i.e., $D = \mathbb{Z}^+$). D includes the stopping value 1.

Step #2 Checking two values is trivial; take 2 and 3.

$$\sum_{i=1}^{2} (-1)^{i-1} i^3 = -\frac{1}{8}(-1)^2 \left(4 * 2^3 + 6 *^2 - 1\right) - \frac{1}{8} = -7$$
$$\sum_{i=1}^{3} (-1)^{i-1} i^3 = -\frac{1}{8}(-1)^3 \left(4 * 3^3 + 6 * 3^2 - 1\right) - \frac{1}{8} = 20$$

To check the stopping value, we check the first value. We clearly see that $f(x) = g(x) = 1$.

Step #3 If n in $\mathbb{Z}^+$ triggers a recursive call, then $n > 0$. The only value used in the call is $n - 1$, which is in $\mathbb{Z}$ and greater than or equal to 0, because it is an integer and $n1 \geq 0$ since $n > 0$.

Step #4 We use the integer n as the counter. When recursion is called the function is called with the value $n - 1$. The counter strictly decreases and the recursion halts.

Step #5 To prove recursion stops, we take the following:

$$\begin{aligned}
f(x) &= f(n-1) + f(n) && (1)\\
&= f(n-1) + (-1)^{n-1}n^3 && (2)\\
&= -\frac{1}{8}(-1)^{(n-1)}\left(4(n-1)^3 + 6(n-1)^2 - 1\right) - \frac{1}{8} + (-1)^{n-1}n^3 && (3)\\
&= -\frac{1}{8}(-1)^{n}\left(4n^3 + 6n^2 - 1\right) - \frac{1}{8} && (4)\\
&= g(x) && (5)
\end{aligned}$$

Because $f(n) = g(n)$, the property is inherited recursively.

Step #6 Since Steps 1–5 have been verified, it follows from the Principle of Recursion that P holds for all values in $\mathbb{Z}^+$, i.e., $f(n) = g(n), \forall n \in \mathbb{Z}^+$

$\square$

Table 1: The results from the explicit form, the series form, and the difference for the first 40 values.

Sum Value	**Explicit Values**	**Difference**
1	1	0
-7	-7	0
20	20	0
-44	-44	0
81	81	0
-135	-135	0
208	208	0
-304	-304	0
425	425	0
-575	-575	0
756	756	0
-972	-972	0
1225	1225	0
-1519	-1519	0
1856	1856	0
-2240	-2240	0
2673	2673	0
-3159	-3159	0
3700	3700	0
-4300	-4300	0
4961	4961	0
-5687	-5687	0
6480	6480	0
-7344	-7344	0
8281	8281	0
-9295	-9295	0
10388	10388	0
-11564	-11564	0
12825	12825	0
-14175	-14175	0
15616	15616	0
-17152	-17152	0
18785	18785	0
-20519	-20519	0
22356	22356	0
-24300	-24300	0
26353	26353	0
-28519	-28519	0
30800	30800	0

```python
def sum_solution(n):
    if n <= 1:
        return 1
    else:
        return (-1)**(n - 1) * n**3 + sum_solution(n - 1)


def explicit_solution(n):
    return -int(
        (1.0 / 8.0) *
        ((-1)**n) *
        (4 * n**3 + 6 * n**2 - 1) +
        1.0 / 8.0)


def main():
    for n in range(1, 40):
        print(sum_solution(n), explicit_solution(n), sum_solution(n) - explicit_solution(n))


if __name__ == "__main__":
    main()
```