Homework #1

Analysis of Algorithms

Illya Starikov

Due Date: September 1^{st}, 2017

Question #1

(a) 4.54 ± 0.05 billion years (`https://en.wikipedia.org/wiki/Age_of_the_Earth`)

(b) 5 billion years (`http://earthguide.ucsd.edu/virtualmuseum/ita/05_3.shtml`)

(c) 13.6 billion years $\pm$ 800 million years (`https://www.space.com/263-milky-age-narrowed.html`)

(d) 13.82 billion years (`http://www.slate.com/blogs/bad_astronomy/2013/03/21/age_of_the_universe_planck_results_show_universe_is_13_82_billion_years.html`)

(e) 7.79 billion years (`https://www.livescience.com/39775-how-long-can-earth-support-life`
`html`)

(f) 3.25 billion years. Guaranteed existential risks, human prevention not possible.

- Asteroid impact
- Extraterrestrial invasion (i.e. , aliens taking over the world)
- Geomagnetic reversal of the North and South pole

Prevention possible.

- Artificial super intelligence enslavement of the man race. Steps are unclear, besides keeping all experiments in a Faraday cage.
- Biotechnology engineering. A bio-engineered virus or disease that gets loose can cause a global pandemic. Steps to prevent are clear.
- Nuclear holocaust. Steps to prevent are clear.

Extinction prevention matters are to become a space-ferrying species, attempt some sort of peace treaty amongst all nations, and to try to prevent global pandemics. (`https://en.wikipedia.org/wiki/Global_catastrophic_risk#Asteroid_impact`)

(g) 10-12 billion years. The sun will become a Red Giant, no longer being able to provide fuel. Will consume several planets along the way. (https://www.forbes.com/sites/startswithabang/2017/01/27/how-our-solar-system-will-end-in-the-far-f #20ea714f4e5d)

> Eventually, about 5–7 billion years down the line, we'll run out of nuclear fuel in the Sun's core, which will cause our parent star to become a Red Giant, engulfing Mercury and Venus in the process. Due to the particulars of stellar evolution, the Earth/Moon system will probably be pushed outwards, and be spared the fiery fate of our inner neighbors.

(h) Either the universe will stop contracting, reach an influx in the gravitational pull on all cosmic bodies, and come back to a singularity; or, heat death, where entropy reaches its max, and all energy is evenly dissipated throughout the universe, freezing everything. The first theory has 1-100 trillion years, the second has no estimate. (http://www.bbc.com/earth/story/20150602-how-will-the-universe-end)

(i) 584.6 billion years. Between 0.585%–58.5%.

Question #2

(a) A standard, 12pt LaTeX document can fit roughly 39 lines of text.

$$\frac{9 \times 10^8}{39} \approx 2.31 \times 10^8 \, \text{sheets}$$

(b) 1500 sheets is approximately \$14.99, before tax. (https://www.amazon.com/Georgia-Pacific-S dp/B00BB5DJU6/ref=sr_1_2?s=office-products&ie=UTF8&qid=1503934559&sr=1-2&keywords=printer+paper)

$$\left(2.31 \times 10^8 \, \text{sheets}\right) \times \frac{\$14.99}{1500 \, \text{sheets}} \approx \$2.31 \times 10^6$$

Assuming a standard filing cabinet can store roughly 15 reams, with 100 sheets in a ream,

$$2.31 \times 10^8 \, \text{sheets} \times \frac{1 \, \text{reams}}{100 \, \text{sheets}} \times \frac{1 \, \text{cabinet}}{15 \, \text{reams}} = 154\,000$$

(c) The monks expected the project to take *15,000 years* by hand, and *3 months* by modern technology. Assuming an average human lifespan of 79 years, approximately 50 years could be used writing.

$$\frac{15\,000 \, \text{years}}{50 \, \text{years}} = 300 \, \text{people}$$

(d) The time allocated to the computer was *3 months*. To print all 3 billion, it would have to print approximately 1120.07 names per second.

(e) I enjoyed the story, and the first thing I did was a mental check to see if the numbers added up. Naturally, they did not, but that did not detract from the story.

Question #3

```python
def alternating_difference(numbers):
    if not numbers:
        return 0

    return numbers[0] - alternating_difference(numbers[1:])
```

Question #4

```python
def f53q(n):
    if  n < 8:
        return ValueError('Value less than 8')
    elif n % 5 == 0:
        return (0, n // 5)
    elif n % 5 == 1:
        return (2, (n - 5) // 5)
    elif n % 5 == 2:
        return (4, (n - 12) // 5)
    elif n % 5 == 3:
        return (1, n // 5)
    else:
        return (3, (n - 9) // 5)
```

Question #5

$$f(1) = 91$$
$$f(-6) = 91$$
$$f(200) = 190$$
$$f(27) = 91$$

A more appropriate, non-recursive function would be as follows:

```python
def f(n):
    if n < 101:
        return 91
    else:
        return n - 10
```

This will always produce the same output. Let us write $f(n)$ as a piecewise-defined function.

$$f(x) = \begin{cases} n - 10 & 100 < n < \infty \\ f(f(n+11)) & -\infty < n \leq 100 \end{cases}$$

We see that, for values less than 100, n will grow until it hits the first case. For values $90 \ldots 200$, we see that n will oscillate until it reaches 101, upon which one final $f(x)$ will be applied.

A more simple explanation would be to insert $n - 10$ into $f(f(n+11))$ to see that $f(x) = f(f(n+1))$ until it reaches 101, 1 past the bounds, and gets $n - 10$ applied to it one more time.

Question #6

The function is copy and pasted from implementation.

```
def A(x, y):
    if x == 0:
        return y + 1
    elif y == 0:
        return A(x - 1, 1)
    else:
        return A(x - 1, A(x, y - 1))
```

The maximum value of my system (Late-2013 MacBook Pro) is roughly `A(3, 5)`. For `A(2, 2)`, there should be 42 438 function calls.

Question #7

```
def gcd2(a, b):
    if a == 0:
        return (b, 0, 1)
    else:
        g, s, t = gcd2(b % a, a)
        return [g, t - (b // a) * s, s]
```

Question #8

```
# defined as list_ to not stomp out the typename _list_
def super_reverse(list_):
    if len(list_) < 2:
        return list_
    else:
        last_element = list_[-1]
        first_element = list_[0]

```

```python
        # if is a list or a tuple, sort that as well
        if isinstance(first_element, list):
            first_element = super_reverse(first_element)
        if isinstance(last_element, list):
            last_element = super_reverse(last_element)

        return [last_element] + super_reverse(list_[1:-1]) + [
    first_element]
```

Question #9

```python
def anagram(string_):
    if string_ == "":
        return [""]

    result = []

    for partial_anagram in anagram(string_[1:]):
        for position in range(len(partial_anagram) + 1) :
            right_side = partial_anagram[position:]
            left_side = partial_anagram[:position]
            original_element = string_[0]

            result += [left_side + original_element + right_side]

    return result
```

Table 1: Results (units in milliseconds)

String Size	**One Recursive Call**	n **Recursive Calls**	**Absolute Difference**	**Percent Difference**
1	0.006	0.004	-0.002	-56.250%
2	0.009	0.009	-0.000	-2.703%
3	0.015	0.026	0.011	42.202%
4	0.036	0.104	0.068	65.367%
5	0.139	0.528	0.389	73.668%
6	0.672	3.124	2.452	78.486%
7	5.312	25.050	19.738	78.794%
8	39.671	204.850	165.179	80.634%
9	332.021	1936.445	1604.424	82.854%
10	3471.336	21813.828	18342.492	84.087%
11	43223.737	267635.574	224411.837	83.850%