HOMEWORK 4
BIG O PERFORMANCES OF LISTS AND DICTIONARIES

For learning purposes, for each of the following problems please try as many methods as possible. I will give credits to the "best" solution.

Problem 1. A number $n > 1$ is called a perfect power if there exists two positive integers m, k such that $k > 1$ and $n = m^k$. For example, 9 is a perfect power because $9 = 3^2$. Similary, 8 is a perfect number because $8 = 2^3$. Write a program to check whether a given number n is a perfect power. What is the big O performance of your program? **Hint:** Suppose that $n = m^k$. Show that $m \leq \sqrt{n}$ and $k \leq \log_2(n)$.

Problem 2. Given two strings, `ransom_note` and `magazine`, write a function named `can_construct(ransom_note, magazine)` that returns True if `ransom_note` can be constructed by using the letters from `magazine`, and False otherwise. For example

```
can_construct("abc", "abcc") #true
can_construct("abc", "abbea") # False
```

What is the time complexity of your algorithm?

Problem 3.

(1) Given a list alist and a specified target value, write a function named `two_sum(alist, target)` that returns True if there exists a pair of elements (with different indices) in the list whose sum equals the target? For example

```
two_sum([2, 2, 3, 5, 8], 8)) # True
two_sum ([2, 2, 3, 5, 8], 16)) # False
two_sum ([2, 2, 3, 5, 8], 4)) # True
```

What is the time complexity of your algorithm?

(2) Can we do better if we know that the list is sorted? What is the time complexity of your algorithm?

Problem 4. Given a list of numbers. Write a function named `non_duplicate(alist)` that returns a new list of elements that appears exactly one in the list.For example

```
non_duplicate([1, 2, 1, 3]) #[2,3]
non_duplicate([1,2, 1, 2]) # []
```

What is the time complexity of your algorithm?

Problem 5. Write a function named latex_checker(file_path) that can validate the syntax of a latex document saved as a text file. For simplicity, the function only need to ensure that all LaTeX environments (i.e., the parts enclosed within \begin{} and \end{} commands) are properly matched and nested. Please test your function with two text files that I uploaded to Moodle. What is the big O performance of your algorithm?

This problem works with text files so it could get tricky. Becareful with the blacklash symbol (\) in latex which turns out to be an escape character when read in Python. You may need to use regular expressions or the string method `startwith` to look for `begin` and `end`.

Problem 6. In this problem, we work with the Hexadecimal system (base 16). Please read more about it here `https://en.wikipedia.org/wiki/Hexadecimal`

(1) Write a function named convert_to_decimal(s) that converts a given a number in the Hexadecimal system to the decimal system. For example

```
convert_to_decimal("2C7")
```

should return 711 since

$$2C7 = 2 \times 16^2 + 12 \times 16^1 + 7 = 711.$$

(2) Write a function named convert_to_hex(n) that converts a given number in the decimal system to its representation in the hexadecimal system. For example

```
convert_to_hex(711)
```

should return 2C7.

Problem 7. Add a method power(self, n) to the ComplexNumber class described in Homework 2. This method should have $O(\log(n))$ performance.