**Regular Languages and Finite State Automata**[**​**](https://xiaohai.wiki/university/cab203-discrete-structure/problem-solving-assignment#regular-languages-and-finite-state-automata)

**Regular Expression**[**​**](https://xiaohai.wiki/university/cab203-discrete-structure/problem-solving-assignment#regular-expression)**(Introduction)**

In this report the regular languages and finite state automata is a Python functionthat takes string s and returns it back, but with all occurrences of a, an, the (regardless of cApitALiSatIOn) replaced with ##, where the number of # characters matches the number in the word being redacted. For example an should be replaced with ##. If your function makes any replacement (and only if it makes a replacement) it should append to the string a single space followed by your student number to the string in the format.

A **regular language** is a common language that can be expressed with a [regular expression](https://brilliant.org/wiki/regular-expressions/) or a deterministic or non-deterministic [finite automata](https://brilliant.org/wiki/finite-state-machines/) or state machine. A **language** is a set of [strings](https://brilliant.org/wiki/strings/) which are made up of characters from a specified alphabet, or set of symbols. Regular languages are a [subset](https://brilliant.org/wiki/sets-subsets/) of the set of all strings. Regular languages are used in parsing and designing programming languages and are one of the first concepts taught in computability courses. These are useful for helping computer scientists to recognize patterns in data and group certain computational problems together — once they do that, they can take similar approaches to solve the problems grouped together. Regular languages are a key topic in computability theory.

The **regular expression** used in the censor function is \b(the|a|an)\b.

Firstly, a capturing group () is used to capture articles in the string. (the|a|an) creates a single group containing one of three words. The will be useful when performing substitutions.

Secondly, according to [RegExp.info](https://www.regular-expressions.info/wordboundaries.html), \b matches at a position that is called a "word boundary". Place \b around the word like this \bword\b allows you to perform a "whole word only" search. So, in order to match all articles (i.e., the, a, an), I place these articles inside \b like this \b(the|a|an)\b.

**Regular Expression (Functions**[**​**](https://xiaohai.wiki/university/cab203-discrete-structure/problem-solving-assignment#regular-expression-functions)**)**

**Regular expression functions** used in censor include re.compile and pattern.sub.

pattern = re.compile(r"\b(the|a|an)\b", flags=re.IGNORECASE)

The re.compile function is used to compile a regular expression into a regex pattern object. We can modify the expression's behavior by setting the flags parameter. In this case, I set the re.IGNORECASE flag on. It is used for performing case-insensitive matching. So that all occurrences of a, an, the regardless of capitalization can be matched.

def replace\_with\_hash(match):  
 return "#" \* len(match.group())  
replaced = pattern.sub(replace\_with\_hash, s)

pattern.sub accepts two arguments. The first argument I pass is a replacement function which can perform dynamic replacements based on the length of the matched *article*. The second argument is the string to be processed.

**Solution**[**​**](https://xiaohai.wiki/university/cab203-discrete-structure/problem-solving-assignment#solution)

This section describes how the functions and syntax are used in solving the problem.

Firstly, compile the regular expression \b(the|a|an)\b with re.IGNORECASE flag set into a pattern. Secondly use the pattern object to substitute all the articles with #. Finally, if the substituted string is the different from the input string (replacement has been made), then append the student ID to the substituted string.

**Linear Algebra**[**​**](https://xiaohai.wiki/university/cab203-discrete-structure/problem-solving-assignment#linear-algebra)

**Introduction**

In this report linear algebra is a python function should return a pair (a,b) where a is the amount of fertiliser of type A required, and b is the amount of fertiliser of type B required. All amounts of fertiliser and nutrients are measured in kg. If there is no solution to the problem your function should instead return None. Note that in this scenario, the farmer cannot add a negative amount of fertiliser to her field, so if a or b is negative then that does not count as a solution to the problem. Likewise, all the arguments to the function will make sense physically, i.e. they will all be non-negative. Mathematically, it is possible for there to be an infinite number of solutions, but this is beyond what we studied in the unit, so such cases can be ignored; they will not be used as test cases.

The problem can be modeled as 2 linear equations with 2 unknowns.

* an is the amount of nitrogen in 1kg of type A fertiliser
* ap is the amount of phosphate in 1kg of type A fertiliser
* bn is the amount of nitrogen in 1kg of type B fertiliser
* bp is the amount of phosphate in 1kg of type B fertiliser
* n is the amount of nitrogen required by the crop
* p is the amount of phosphate required by the crop
* a is the amount of type A fertiliser required
* b is the amount of type B fertiliser required

**Linear Algebra Functions(Mathematical Description)**

We want to solve the following two linear equations. a and b are the unknowns.

an \* a + bn \* b = n

ap \* a + bp \* b = p

For a and b. We can represent these two equations in vectors.

Then a and b can be solved by finding the inverse of the matrix and multiply it with

Note : If det , then it will not have an inverse ( a singular matrix). So there is no solution to the problem.

Secondly, because the farmer cannot add a negative amount of fertiliser, so if a or b is negative then that is not considered as a solution.