Algorithms in Python

1. Imagine that you wish to make change for a sum Q with coins of denominations: d_1, d_2, ..., d_n. Write a function makeChange(Q, D) that gives change for Q with denominations available in the given list. Make sure that you use fewest possible number of coins. You are allowed to use as many number of coins of each denomination as needed.

   For example:
   Let $Q = 10$ and $D = [1, 2, 3]$ Q can be achieved with any of the following combinations:

   1. $[1, 1, 1, 1, 1, 1, 1, 1, 1, 1]$
   2. $[2, 2, 2, 2, 2]$
   3. $[3, 3, 3, 1]$

   etc.

   The third option is the correct answer since it has the smallest number of coins.

2. You are given n jugs of capacities c_1, c_2, ..., c_n in litres. Design a program that takes a list of integers, corresponding to these capacities, and another integer c. As output, your algorithm should print out the steps to follow such that one of the n jugs has c litres of water. In the process you are allowed to perform only the following types of operations:

   1. fill (j): Fills the jth jug to its capacity. As a result, j will have c_j litres of water.
   2. empty(j): Empties jth jug. As a result, j will have have 0 litres of water.
   3. transfer (i, j): Transfers the contents of ith jug to the jth jug. Consider it a part of the exercise to figure out all the cases for this operation.

   For example:

```
$ ./jugs 3 4 5 2
(0, 0, 0)
(0, 0, 5)
(3, 0, 2)
(0, 0, 2)
(0, 0, 2)
```

```
$ ./jugs 4 5 2
(0, 0)
(0, 5)
(4, 1)
(0, 1)
(1, 0)
(1, 5)
(4, 2)
(0, 2)
```

3. Consider two numbers a and b. We can create any number of *power-products* of a and b. For example: a^0b^0, a^1b^0, a^0b^1, a^1b^1, a^2b^0, a^2b^1, a^2b^2 ... With $a = 2$ and $b = 3$, the products would be $2^0 3^0 = 1$, $2^1 3^0 = 2$, $2^0 3^1 = 3$, $2^1 3^1 = 6$, $2^2 3^0 = 4$, ... and so on.

   Write a function `power_products(a, b, n)` which computes the first n smallest power-products of a and b where a and b are integers greater than 1, and n is an integer greater than or equal to 1.