1. Carry out the following multiplication of two 4-digit numbers using the grade school algorithm: 1234 · 5678.

Using the grade school grid algorithm:

1234 * 5678	=	7006652			
	5000	600	70	8	
1000	5000000	600000	70000	8000	
200	1000000	120000	14000	1600	
30	150000	18000	2100	240	
4	20000	2400	280	32	
	6170000	740400	86380	9872	7006652

2. Now answer the following questions:
   (a) How many “steps” did this computation take?
   Took 17 steps. 16 to multiply each of the grid possibilities, 1 to add up all the respective numbers
   (b) How do you define a step? In particular, which operations do you need to consider as steps?
   I define a step as either one multiplication or a formula with sums.
   (c) What is the number of steps in a given algorithm a good metric for?
   It’s efficiency and how many resources it will demand from the computer. It is also good to measure the speed of the algorithm.
   (d) Can you use a while loop to perform this multiplication? If so, how? If not, why not? What about a recursive approach?
   Yes, we can use while loops. It would have to keep track of the number of digits in each number (4 in both cases) and separate them (in a list for example). Then using a counter that starts at the number’s length - 1 (in this case 3 for both) we add *x* amount of 0s to each digit so that we get 1000, 200, 30, 4, and 5000, 600, 70, 8. Then we can use another while loop that makes each number inside the first list multiply itself with each respective number in the second list, we can use another counter to keep track of when we run out of options and move on to the next number (e.g. 5000*1000, 5000*200, 5000*30, 5000*4). Finally, we just sum everything.

   We can also use a recursive approach. It would look something like this:

```
Def grid_multiply(input):
        Breakup_numbers
        If NOT broken_up:
                grid_multiply(input 1:)
        Else:
                …
```