1. Print the following pattern:

   (a)

   For n = 4:

```
4 4 4 4 4 4 4
4 3 3 3 3 3 4
4 3 2 2 2 3 4
4 3 2 1 2 3 4
4 3 2 2 2 3 4
4 3 3 3 3 3 4
4 4 4 4 4 4 4
```

   (b) For n=3:

```
    1
  1   1
1   2   1
```

2. Developers write function and variable names in Camel Case. You are given a string, S, written in Camel Case.
   FindAllTheWordsContainedInIt.

   Sample input: IAmACompetitiveProgrammer

   Sample output:

```
I
Am
A
Competitive
Programmer
```

3. Study about Walrus Operator(“:=”)

4.
```
funny_list = [
    "some",
    "very",
    "big",
    "list",
    "that"
    "consists",
    "of",
    "exactly",
    "ten",
    "words"
]
len(funny_list)
```

Output: 9

Wait!! What?? Yes, so figure out why this happened.
Hint: Check the list again.

5. Statement 1: 'world' is not None
   Statement 2: 'world' is (not None)
   Are these statements equivalent? find the reason for your answer.

6. Classes and inheritance:
   Create a class AnimalKingdom with the following properties and methods.
   Properties:
   - Population(integer)
   - Name(string)
   - Type(string): Can be herbivore, carnivore or omnivore
   - Number of Limbs(integer)
   - Habitat(string)
   - is_alive(boolean)

   Methods:
   - die(self) : Method should alter is_alive and population
   - eat(self, animal): Method for an animal to eat other animals.
   -

   Create three other classes Herbivore, Carnivore, Omnivore which inherit AnimalKingdom. Derived classes should have a property for recording their own population. E.g.

   print(Carnivore.population)

   should only print the population of animals which are instance of Carnivore class.
   Similarly upon dying or being eaten by other animal populations of these derived classes should update accordingly.

   Bonus:
   - Creating or killing an instance of derived classes should automatically update the population of AnimalKingdom
   - Write a check for instance of Herbivore class that it can not eat other animals.
   - Update *__repr__* method to print name, number of limbs, whether alive or dead.

7. Explore the naming conventions in python for:
   - Single Leading Underscore: _var
   - Single Trailing Underscore: var_

- Double Leading Underscore: __var
- Double Leading and Trailing Underscore: __var__
- Single Underscore: _

8.
```
for x in range(10):
    if x == 6:
        print(x, ': for x inside loop')
print(x, ': x in global')
```

   Execute the given line of code, and tell the reason for the output you get. Does x behave like a global variable or a local variable defined inside for loop? Find a valid reason to support your answer.

9.
```
>>> 'a'.split()
['a']

# is same as
>>> 'a'.split(' ')
['a']

# but
>>> len(''.split())
0

# isn't the same as
>>> len(''.split(' '))
1
```

   Don’t know why?? Then check the docs. Now, are you able to answer this why?

10. Take as input S, a string. Write a function that replaces every odd character with the character having just higher ascii code and every even character with the character having just lower ascii code. Print the value returned.

    Sample input: abcg
    Sample output: badf