```
from typing import List, Union


def depth(obj: Union[list, int]) -> int:
    """ Return the depth of obj.

    >>> s = [[1, 2],[3, 4],[[5]]]
    >>> depth(s)
    3
    >>> depth(1)
    0
    """
    pass


def contains(obj: Union[list, int], item: int) -> bool:
    """ Return True iff the item is in the list of obj or is
    equals to the obj. Otherwise return False.

    >>> s = [[1, 2],[3, 4],[[5]]]
    >>> contains(s, 5)
    True
    """
    pass

def count(obj: Union[list, int]) -> int:
    """ count the number of int in nested list.

    >>> s = [[1, 2],[3, 4],[[5]]]
    >>> count(s)
    5
    """


    pass

def count_greater(obj: Union[list, int], n: int) -> int:
    """ count the number of int in nested list that is greater than n.

    >>> s = [[1, 2],[3, 4],[[5]]]
    >>> count_above(3)
    2
    """
    pass


def equal(obj1, obj2):
    """Return whether two nested lists are equal, i.e., have the same
value.

    Note: order matters.
```

```
    @type obj1: int | list
    @type obj2: int | list
    @rtype: bool

    >>> equal(17, [1, 2, 3])
    False
    >>> equal([1, 2, [1, 2], 4], [1, 2, [1, 2], 4])
    True
    >>> equal([1, 2, [1, 2], 4], [4, 2, [2, 1], 3])
    False
    >>> equal([], 2)
    False
    >>> equal(2, 3)
    False
    """
    pass


def gather(obj: Union[list, int]) -> list:
    """ Return a list of int in nested list.

    >>> s = [[1, 2],[3, 4],[[5]]]
    >>> gather(s)
    [1, 2, 3, 4, 5]
    """
    pass

def gather_smaller(obj: Union[list, int], n: int) -> list:
    """ Return a list of int in nested list that is greater than n.

    >>> s = [[1, 2],[3, 4],[[5]]]
    >>> count_above(3)
    [1, 2]
    """
    pass

def avg(obj: Union[list, int]) -> int:
    """ Return the avg of all the numbers in obj

    hint: you may use helper function for this.

    >>> s = [[1,2],[3,4],[[5]]]
    >>> avg(s)
    3.0
    """
    pass


def all_perm(s):
```

```
    """ Return the all perm list of this string.

    >>> all_perm('abc')
    ['abc', 'bac', 'bca', 'acb', 'cab', 'cba']
    """


def floor_to_int(lst: Union[list, float]) -> None:
    """ Floor each float in the nest list to the int.

    >>> lst = [3.3, [2.2, [9.9]]]
    >>> floor_to_int(lst)
    >>> lst
    [3, [2, [9]]]
    """
    pass


def list_level(obj:List[Any], d:int) -> List:
    """

    Return the non-list elements at a particular level.

    @param list obj: possibly nested list
    @param int d: The level to print out
    @rtype: List

    >>> list_ = [1, [2, [3, 4], 5], 2]
    >>> list_level(list_, 2)
    [2, 5]
    >>> list_level(list_, 3)
    [3, 4]

    """


def list_above_level(obj:List[Any], d:int) -> List:
    """

    Return the non-list elements above particular level.

    @param list obj: possibly nested list
    @param int d: The level to print out
    @rtype: List

    >>> list_ = [1, [2, [3, 4], 5], 2]
    >>> list_level(list_, 3)
    [1, 2, 5, 2]
    """
```

```
if __name__ == "__main__":
    import doctest
    doctest.testmod()
```