Pergunta 1

Write a function called replace_rec(old_ch, new_ch, astring) that replaces all occurrences of a specified character, old_ch, with another specified string, new_ch, in the given string astring. This function is case sensitive.

You must use recursion; you cannot use for loops, while loops or the string replace.
Por exemplo:

Teste	Resultado
print(replace_rec("e", "_", "these are the best days"))	th_s_ ar_ th_ b_st days
print(replace_rec(",", ".", "1,2,3,4,5,6,7,8,9"))	1.2.3.4.5.6.7.8.9
print(replace_rec("M", "#", "NY^T&^MM#M%f*zv#u"))	NY^T&^####%f*zv#u
print(replace_rec("s", "@", "CaseSensitiveTest"))	Ca@eSen@itiveTe@t

Pergunta 2

Given a nested list alist (i.e., a list which may contain lists which themselves may contain other lists, and so on), write a recursive Python function flatten(alist) that returns a single list with each of the non-list elements, in order of occurrence. This process is known as list flattening.

You must use recursion; you cannot use for loops, while loops.
Por exemplo:

Teste	Resultado
print(flatten([]))	[]
print(flatten(['Hello', [2, [[], False]], [True]]))	['Hello', 2, False, True]
print(flatten([[1,[2,3]], [4,[[5]],6], [7,8,9]]))	[1, 2, 3, 4, 5, 6, 7, 8, 9]
print(flatten([1]))	[1]

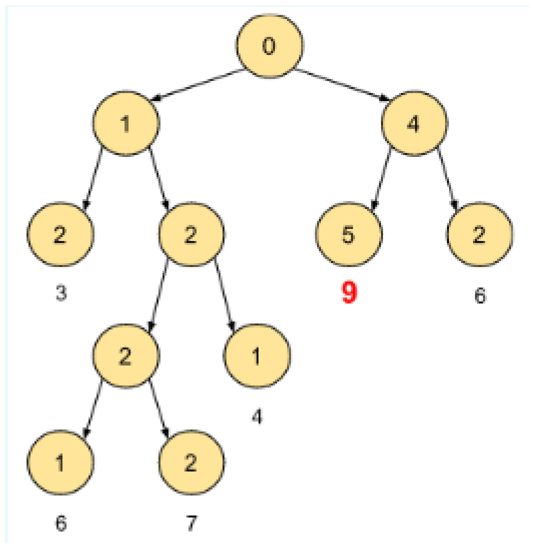
Pergunta 3

Write a recursive Python function `max_path(tree)` that given a tree, computes the value of the path that produces the highest value by summing the values of each node, starting from the root and ending in a leaf (a node without descendants).

A tree is either:

1. a single integer;
2. a triple (left, value, right) where value is an integer value of the node and left and right are the two trees.

For example, considering the tree `((2, 1, ((1, 2, 2), 2, 1)), 0, (5, 4, 2))` the function should return 9.



You must use recursion; you cannot use for loops, while loops.
Por exemplo:

Teste	Resultado
print(max_path(((2, 1, ((1, 2, 2), 2, 1)), 0, (5, 4, 2))))	9
print(max_path((1, 3, 2)))	5
print(max_path(((1, 1, 3), 0, 3)))	4
print(max_path(((2, 3, (4, 5, 2)), 0, (7, 1, 3))))	12

Pergunta 4

Let dirs be a recursively-defined nested tuple and list, representing a directory tree containing an arbitrary number of directories, sub-directories and files.

Consider the following example:

```
dirs = ("home", [
        ("Documents", [
          ("FP", ["lists.txt", "recursion.pdf", "functions.ipynb"]),
          ("Python", ["hello_world.py", "readme.md"])
        ]),
        ("Downloads", [
          ("Movies", [
            ("TV Series", ["BreakingBad.mp4", "TheBigBangTheory.avi"]),
            "1.avi", "22", "001.mp4"
          ])
        ]),
        "tmp.txt", "page.html"])
```

In the above example, each directory is defined by a pair with the name of that directory (home, Documents, FP, etc) and the list of the content of the directory. The list of the elements contain either strings, which are filenames inside that directory, or tuples, which contain a sub-directory.
Write a recursive Python function `file_finder(dirs, file_name)` that returns the full path for the first occurrence of a file `file_name` (given as a string), or `None` if it is not in the directory tree `dirs`.
Inside a directory, the function opens the sub-directories before looking at the files. The full path of a file includes the slash-separated names of all the directories that contain it. Therefore, the full path of "BreakingBad.mp4" is "home/Downloads/Movies/TV Series/BreakingBad.mp4".
For example:

1. `file_finder(dirs, 'Documents')` returns: `None` (because Documents is a sub-directory not a file)
2. `file_finder(dirs, 'recursion.pdf')` returns the string: `"home/Documents/FP/recursion.pdf"`

Por exemplo:

Teste
print(file_finder(("home", [("Documents", [("FP", ["lists.txt", "recursion.pdf", "functions", "tmp.txt"]), ("Python", ["hello_world.py", "readm
print(file_finder(("home", [("Documents", [("FP", ["lists.txt", "recursion.pdf", "functions", "tmp.txt"]), ("Python", ["hello_world.py", "readm
print(file_finder(("home", [("Documents", [("FP", ["lists.txt", "recursion.pdf", "functions", "tmp.txt"]), ("Python", ["hello_world.py", "readm
print(file_finder(("home", [("Documents", [("FP", ["lists.txt", "recursion.pdf", "functions", "tmp.txt"]), ("Python", ["hello_world.py", "readm

Resultado
`None`
`home/page.html`
`home/Documents/Python/hello_world.py`
`home/Downloads/Movies/22`

Pergunta 5

Write a function soup(matrix, word) that, given a matrix of letters, returns the first location of the word in the matrix, or None if not found.

For example, let's say we have the following matrix and are trying to find the word "PORTO".

	1	**2**	**3**	**4**	**5**	**6**
A	X	R	Z	B	H	A
B	K	A	S	I	G	O
C	J	O	T	C	A	N
D	F	S	R	H	T	U
E	D	P	O	O	X	F
F	Z	B	B	W	F	S

Then the function returns "E2" because the word starts in line=E, column=2.

Notice that the words can use any cardinal direction: north, east, south, west.

All letters are given in upper-case and the function returns the first occurrence of the word using lexicographical order (i.e. if the word can be found in "A4" and "B2", then it returns "A4"). You may want to use chr() and ord() to have the line as a letter.

Por exemplo:

Teste
print(soup([['X', 'R', 'Z', 'B', 'H', 'A'], ['K', 'A', 'S', 'I', 'G', 'O'], ['J', 'O', 'T', 'C', 'A', 'N'], ['F', 'S', 'R', 'H', 'T', 'U'], ['D', 'P', 'O', 'O', 'X', 'F'], ['Z', 'B', 'B', '
print(soup([['X', 'R', 'Z', 'B', 'H', 'A'], ['K', 'A', 'S', 'I', 'G', 'O'], ['J', 'O', 'T', 'C', 'A', 'N'], ['F', 'S', 'R', 'H', 'T', 'U'], ['D', 'P', 'O', 'O', 'X', 'F'], ['Z', 'B', 'B', '
print(soup([['X', 'R', 'Z', 'B', 'H', 'A'], ['K', 'A', 'S', 'I', 'G', 'O'], ['J', 'O', 'T', 'C', 'A', 'N'], ['F', 'S', 'R', 'H', 'T', 'U'], ['D', 'P', 'O', 'O', 'X', 'F'], ['Z', 'B', 'B', '
print(soup([['X', 'R', 'Z', 'B', 'H', 'A'], ['K', 'A', 'S', 'I', 'G', 'O'], ['J', 'O', 'T', 'C', 'A', 'N'], ['F', 'S', 'R', 'H', 'T', 'U'], ['D', 'P', 'O', 'O', 'X', 'F'], ['Z', 'B', 'B', '

Resultado
E2
C2
B5
D5