Exercise (H4.1)

Write a function that prints the elements of a given tuple by their ‘level’ (using only tuples and lists, no recursion).

E.g., given the tuple (1.1, 1.2, (2.1, 2.2), 1.3, 1.4, (2.3, (3.1, 3.2), 2.4), 1.5), the function prints 1.1 1.2 1.3 1.4 1.5 2.1 2.2 2.3 2.4 3.1 3.2

Exercise (H4.2)

Write a function that given an ordered list of numbers L and another number n, inserts n in L keeping the order. Implement an in-place modification.

E.g.,

L = [1,3,4,5,6]

n = 2

add_in_order(L,n)

print(L) prints [1,2,3,4,5,6]

Exercise (H4.3)

Write a function that removes the duplicates from a given list. Implement an in-place modification.

E.g.,

L = [1,3,1,5,3]

del_dup(L)

print(L) prints [1,3,5]

Exercise (H4.5)

Write a function that concatenates index-wise two given lists of strings of equal length.

E.g.,

```
list1 = ["M", "na", "i", "An"]

list2 = ["y", "me", "s", "gelo"]

list3 = concatenate_indexwise(list1, list2)
```

print(list3) prints ["My", "name", "is", "Angelo"]

Exercise (H4.6)

Write a program to play ‘four-in-a-row’. Given a 4x4 board, two users take turns in filling in the 16 positions. At each turn, a user is asked to fill in (using “X” or “ O”) the next available position in their prefered row. The user that fills in an entire row first wins the game.

At each turn, the program should print the current state of the board and ask the users to insert the move (indicating the row to be filled in).

Exercise (H4.6)

E.g.,

```
1: X O
2: XXO
3: O
4:
Insert move for “X”:  3
```

```
1: X O
2: XXO
3: OX
4:
Insert move for “O”:
```

…

Exercise (H4.6)

Write a program to play ‘four-in-a-row’. Given a 4x4 board, two users take turns in filling in the 16 positions. At each turn, a user is asked to fill in (using “X” or “ O”) the next available position in their prefered row. The user that fills in an entire row first wins the game.

At each turn, the program should print the current state of the board and ask one of the users to insert the move (indicating the row to be filled in).

The program should do the necessary checks (when the user does not input a number between 1 and 4, when the indicated row is already full etc), and take actions accordingly for instance by aborting the execution or by asking a new move etc.

Exercise (H4.6)

Tip: represent the game with a tuple of 4 lists of strings, each corresponding to the current state of a game row (initially all empty). Define a print_game function that given this type of tuple, prints out the game grid corresponding to the lists in the tuple.

Exercise (H4.7)

Write a function that given a nested list (which may contain lists containing lists etc), prints the elements of the list. You need to first flatten the list

E.g., given the list ["a", "b", ["c", ["d", "e", ["f", "g"], "k"], "l"], "m", "n"], the function prints

"a" "b" "c" "d" "e" "f" "g" "k" "l" "m" "n"

Tip: use the list like a stack.