Name-Surname:Chiho Li ID: 64011378

Introduction to Computers and Programming, SE Programme

Homework #13

22nd November 2021

1. Write a recursive function to traverse and print a binary tree.

For example, **print_btree([1, [[11, [111, 112]], [12, [121, [122, [1221, 1222]]]]], 0)**, it will print

```
1
. 11
. . 111
. . 112
. 12
. . 121
. . 122
. . . 1221
. . . 1222
```

The tree pattern of the first parameter is: [node, [left sub-tree, right sub-tree]]. The second parameter is a depth that the recursive function will use to count the number of dots when it prints out the result. At the beginning, this parameter will be set to zero and it will get increase incrementally when the function makes a recursive call to itself.

1 . 11 . . 111 . . 112 . 12 . . 121 . . 122 . . . 1221 . . . 1222	Depth = 0, Root, Children = [[11, ...], [12, ...]] Depth = 1, Left child, Children = [111, 112] Depth = 2, Left child, Leaf Depth = 2, Right child, Leaf Depth = 1, Right child, Children = [121, [122, ...]] Depth = 2, Left child, Leaf Depth = 2, Right child, Children = [1221, 1222] Depth = 3, Left child, Leaf Depth = 3, Right child, Leaf

2. Let f be the function on natural numbers defined recursively as follows:

f(0) = 0,
f(n) = 2 * f(n/2) + 1, if n > 0 and n is even
f(n) = 0, if n > 0 and n is odd

Write a Python function **display_f(n)**, which given an integer **n** ≥ 0, prints out the value of f(0), f(1), ..., f(n).

```python
def display_f(n):
    if n == 0:
        return n
    elif n > 0 and n % 2 == 0:
        return display_f()
    elif n > 0 and n % 2 != 0:
        return display_f()

print(display_f())
```

3.1 Write a Python recursive function **perm2(t)** which, given a tuple **t** of numbers, prints out all possible **pairs** of distinct numbers in **t**. Each pair that is printed out may not contain duplicating numbers.

For example,
>>> **perm2((1,2,3))**
(1,2) (1,3) (2,1) (2,3) (3,1) (3, 2)

3.2 Write a Python recursive function **perm3(t)** which, given a tuple **t** of numbers, prints out all possible **triples** of distinct numbers in **t**. Each triple that is printed out may not contain duplicating numbers.

For example,
>>> **perm3((1, 2, 3, 4))**
(1,2,3) (1,2,4) (1,3,2) (1,3,4) (1,4,2) (1,4,3)
(2,1,3) (2,1,4) (2,3,1) (2,3,4) (2,4,1) (2,4,3)
(3,1,2) (3,1,4) (3,2,1) (3,2,4) (3,4,1) (3,4,2)
(4,1,2) (4,1,3) (4,2,1) (4,2,3) (4,3,1) (4,3,2)

3.3 Write a Python recursive function **perm(t, n)** which, given a tuple **t** of numbers and a number **n** ≥ 0, prints out all possible **n**-tuples of numbers in **t**. Each tuple that is printed out may not contain duplicating numbers. You may assume that **n** is no greater than the number of elements in **t**.

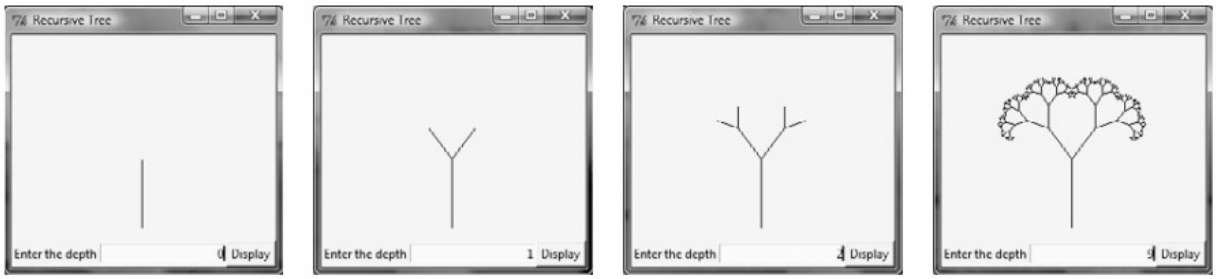
perm(t,2) should produce the same result as **perm2(t)**.
perm(t,3) should produce the same result as **perm3(t)**.

For example,
>>> **perm((1, 2, 3), 3)**
(1,2,3) (1,3,2) (2,1,3) (2,3,1) (3,1,2) (3,2,1)

send help

4. Write a recursive program to solve the tower of Hanoi and draw an animation of it.

5. Write a program to display a recursive tree, as shown below.



(a) (b) (c) (d)

```
from turtle import *

def tree(length, order):
    if length < (length / order):
        return
    forward(length)
    left(45)
    tree(length * 0.5, length / order)
    right(90)
    tree(length * 0.5, length / order)
    left(45)
    backward(length)
    return

left(90)
tree(100, 4)
speed(0)
done()
```