



## **Homework #13**

**01286121 Computer Programming  
Software Engineering Program,  
Department of Computer Engineering,  
School of Engineering, KMITL**

By

66011149 Phatthadon Sornplang

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

```
def print_btree(tree, dpt=0):
    if isinstance(tree,int):
        print("." * dpt + str(tree))
    else:
        print("." * dpt + str(tree[0]))
        if len(tree) >= 1:
            for child in tree[1]:
                print_btree(child,dpt + 1)
```

```
tree = [1,[[11, [111, 112]],[12, [121, [122, [1221, 1222]]]]]]
```

```
print_btree(tree)
```

```
phatt@Macbook_Pro MINGW64 ~/Desktop/Code Files/Python/Computer Programming (Python)/13
$ C:/Users/phatt/AppData/Local/Programs/Python/Python311/python.exe "c:/Users/phatt/Desk
1
.11
..111
..112
.12
..121
..122
...1221
...1222
```

2. Write a Python function display\_f(n), which given an integer  $n \geq 0$ , prints out the value of  $f(0)$ ,  $f(1)$ ,...  $f(n)$ .

```
def f(n):
    if n == 0:
        return 0
    if n > 0 and n % 2 == 1:
        return 0

    return 2 * f(n // 2) + 1
```

```
def display_f(n):
    for i in range(n+1):
        print(f(i))
```

```
display_f(5)
```

```
phatt@Macbook_Pro MINGW64 ~/Desktop/Code Files/Python/Computer Programming (Python)/13
$ C:/Users/phatt/AppData/Local/Programs/Python/Python311/python.exe "c:/Users/phatt/De
0
0
1
0
```

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.

```
def perm2(t):
    for i in range(len(t)):
        for j in range(len(t)):
            if i != j:
                print((t[i], t[j]), end = " ")
```

```
phatt@Macbook_Pro MINGW64 ~/Desktop/Code Files/Python/Computer Programming (Python)/13
$ C:/Users/phatt/AppData/Local/Programs/Python/Python311/python.exe "c:/Users/phatt/Desktop/perm2.py"
(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.

```
def perm3(t):
    for i in range(len(t)):
        for j in range(len(t)):
            for k in range(len(t)):
                if i != j and i != k and j != k:
                    print((t[i], t[j], t[k]), end = " ")
```

```
phatt@Macbook_Pro MINGW64 ~/Desktop/Code Files/Python/Computer Programming (Python)/13
$ C:/Users/phatt/AppData/Local/Programs/Python/Python311/python.exe "c:/Users/phatt/Desktop/perm3.py"
(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)
```

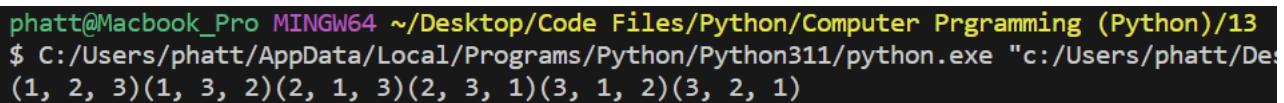
3.3. Write a Python recursive function `perm(t, n)` which, given a tuple `t` of numbers and a number `n ≥ 0`, prints out all possible `n`-tuples of numbers in `t`. Each tuple that is printed out may not contain duplicating numbers. You may assume that `n` is no greater than the number of elements in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

```
def perm(t, n):
    if n == 0:
        return [()]

    if not t:
        return []

    result = []
    for i in range(len(t)):
        rest = perm(t[:i] + t[i+1:], n-1)
        for p in rest:
            result.append((t[i],) + p)
    return result

def print_perm(t, n):
    for p in perm(t, n):
        print(p, end = " ")
```



```
phatt@Macbook_Pro MINGW64 ~/Desktop/Code Files/Python/Computer Programming (Python)/13
$ C:/Users/phatt/AppData/Local/Programs/Python/Python311/python.exe "c:/Users/phatt/De
(1, 2, 3)(1, 3, 2)(2, 1, 3)(2, 3, 1)(3, 1, 2)(3, 2, 1)
```

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.

```
import turtle as t

t.left(90)
t.tracer(0)

def tree(input):
    if input < 10:
        return
    else:
        t.fd(input)
        t.left(30)
        tree((3*input)/4)
        t.right(60)
        tree((3*input)/4)
        t.left(30)
```

```
t.back(input)
```

```
tree(100)
```

```
t.done()
```

