CSCI-SHU 210 Data Structures

Recitation 3 Recursion

Problem 1
Tower of Hanoi

Complete function hanoi( ) in hanoi.py, so the disks move correctly.

** You also need to print total number of moves.

Starting point of Tower of Hanoi is here.

Problem 2
Binary Search

Complete function binary_search( ): this function uses a binary search to determine whether an ordered list contains a specified value.

We will implement two versions of binary search:

1. Recursive
2. Iterative

Starting point of Binary Search problem is here.

Problem 3
Palindrome (Recursive version)

Implement function palindrome( ): this function assesses whether an input String is indeed a palindrome.

Starting point of Palindrome problem is here.

Problem 4
All Possible Combinations problem

Implement a recursive approach to show all the teams that can be created from a group (out of n things choose k at a time). Implement the recursive showTeams( ), given a group of players, and the size of the team, display all the possible combinations of players.

Starting point of show team problem is here.

Example Input:

```
players = [“Dey”, “Ruowen”, “Josh”, “Kinder”, “Mario”, “Rock”, “LOL”] # 7
 players
show_team(players, 4, [], 0) # Pick 4 from 7
```

should output:

```
['Kinder', 'Mario', 'Rock', 'LOL']
['Josh', 'Mario', 'Rock', 'LOL']
['Josh', 'Kinder', 'Rock', 'LOL']
['Josh', 'Kinder', 'Mario', 'LOL']
['Josh', 'Kinder', 'Mario', 'Rock']
['Ruowen', 'Mario', 'Rock', 'LOL']
['Ruowen', 'Kinder', 'Rock', 'LOL']
['Ruowen', 'Kinder', 'Mario', 'LOL']
['Ruowen', 'Kinder', 'Mario', 'Rock']
['Ruowen', 'Josh', 'Rock', 'LOL']
['Ruowen', 'Josh', 'Mario', 'LOL']
['Ruowen', 'Josh', 'Mario', 'Rock']
['Ruowen', 'Josh', 'Kinder', 'LOL']
['Ruowen', 'Josh', 'Kinder', 'Rock']
['Ruowen', 'Josh', 'Kinder', 'Mario']
['Professor Day', 'Mario', 'Rock', 'LOL']
['Professor Day', 'Kinder', 'Rock', 'LOL']
['Professor Day', 'Kinder', 'Mario', 'LOL']
['Professor Day', 'Kinder', 'Mario', 'Rock']
['Professor Day', 'Josh', 'Rock', 'LOL']
['Professor Day', 'Josh', 'Mario', 'LOL']
['Professor Day', 'Josh', 'Mario', 'Rock']
['Professor Day', 'Josh', 'Kinder', 'LOL']
['Professor Day', 'Josh', 'Kinder', 'Rock']
['Professor Day', 'Josh', 'Kinder', 'Mario']
['Professor Day', 'Ruowen', 'Rock', 'LOL']
['Professor Day', 'Ruowen', 'Mario', 'LOL']
['Professor Day', 'Ruowen', 'Mario', 'Rock']
['Professor Day', 'Ruowen', 'Kinder', 'LOL']
['Professor Day', 'Ruowen', 'Kinder', 'Rock']
['Professor Day', 'Ruowen', 'Kinder', 'Mario']
['Professor Day', 'Ruowen', 'Josh', 'LOL']
['Professor Day', 'Ruowen', 'Josh', 'Rock']
['Professor Day', 'Ruowen', 'Josh', 'Mario']
['Professor Day', 'Ruowen', 'Josh', 'Kinder']
```

Problem 5

Use Turtle module draw a Tree

Starting point of Tree problem is here.

Turtle module is a python built in module. Turtle module draws lines by moving the cursor.

```
turtle functions explained:
import turtle

t = turtle.Turtle()    # Initialize the turtle

t.left(30)             # The turtle turns left 30 degrees
t.right(30)            # The turtle turns right 30 degrees
t.forward(20)          # The turtle moves forward 20 pixels, leave a line on the path.
t.backward(30)         # The turtle moves backward 30 pixels, leave a line on the path.

... and more! In this lab, that's all we need.
```

With the mind set of recursion, let's break down this problem.

If the branch is too small, stop.

Otherwise, we should branch two new branches (two recursions, two smaller problems)

Go forward
Turn left
Recursion a smaller branch
Turn right
Recursion another smaller branch
Go back to where we started