CSCI-SHU 210 Data Structures

Assignment 3 Recursion

Problem 1

What is the running time of myRecFunc(n)?

```
def myRecFunc(n):
      print(n, ":", end = "")
      if (n < 1): return;
      myRecFunc(n // 2)
```

Problem 2

x to the power of n.

Implement power(x,n) recursively. n could be any integer (positive, negative, or zero). In the text book, power(x,n) python program (Page Number 173 Code Fragment 4.12) only handles non-negative integers. You may modify that program so that it can handle negative integers too.

- Parameter 1: base
- Parameter 2: exponent
- Return: base to the power of exponent.

** Of course, you cannot use "**" power operator!

Example 1:

```
print(power(5, 2))  # 5^2 = 25
>>> 25
```

Example 2:

```
print(power(4, -1))  # 4^(-1) = 0.25
>>> 0.25
```

Problem 3

Rewrite the following recursive method using iteration instead of recursion.

```
def recur(n):
      if (n < 0):
            return -1
      elif (n < 10):
            return 1
      else:
            return (1 + recur(n // 10))
```

Problem 4
Element Uniqueness Problem

Consider the Unique3 (Recursive Element Uniqueness Problem) program from the text book (Page Number 165 Code Fragment 4.6). Worst case runtime for this program could be O(2^n).

Implement an efficient recursive function unique(S) for solving the element uniqueness problem, which runs in time that is at most O(n^2) in the worst case without using sorting.

- parameter1: List
- ** Add additional parameters if you want
- Return True if there are no duplicate elements in List S.
- Return False otherwise

Example 1:

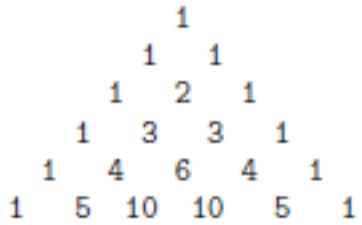
```
print(unique([1,7,6,5,4,3,1]))
>>> False
```

Example 2:

```
print(unique([9,4,3,2,1,8]))
>>> True
```

Problem 5
Pascal's Triangle

In mathematics, Pascal's triangle is a triangular array of the binomial coefficients. Implement a recursive function pascal(N) to display a triangle like the following:



Implement recursive function pascal(N), which generates list of sublists, each sublist contains a level of Pascal values. Try to find base cases and find the relationship between two levels. Can we generate the next level from the previous level?

- parameter1: number of levels
- return: list of sublists. Those lists contains pascal values.

Example:

```
print(pascal(5))
```

```
[[1], [1, 1], [1, 2, 1], [1, 3, 3, 1], [1, 4, 6, 4, 1]]
```

Also implement an iterative function in main( ): Calls pascal(N), use the returned list to display triangle.

- parameter: None
- return: None
- print: the pascal triangle

Example:

```
def main():
        result = pascal(5)
        # Your code need to print result nicely
```

Printed:

```
    1
   1 1
  1 2 1
 1 3 3 1
1 4 6 4 1
```

Problem 6
All Possible Combinations

Complete function generateBillboard(S): this function enumerates and displays all possible casts (combinations) of celebrities from a specified list that can appear on the billboard of a movie with a given number of lead roles.

- parameter1: Input List
- parameter2: Integer, number of celebrities appearing together.
- parameter3: helper List for recursion enumerating
- parameter4: helper Integer for recursion indexing
- Return: Nothing
- Print: All possible Combinations

Example:

```
casts = ["Johnny Depp", "Al Pacino", "Robert De Nir
o", "Kevin Spacey", "Denzel Washington", "Russell C
rowe", "Brad Pitt"]

generateBillboard(casts, 2, [], 0)

['Russell Crowe', 'Brad Pitt']
['Denzel Washington', 'Brad Pitt']
['Denzel Washington', 'Russell Crowe']
['Kevin Spacey', 'Brad Pitt']
['Kevin Spacey', 'Russell Crowe']
['Kevin Spacey', 'Denzel Washington']
['Robert De Niro', 'Brad Pitt']
['Robert De Niro', 'Russell Crowe']
['Robert De Niro', 'Denzel Washington']
['Robert De Niro', 'Kevin Spacey']
['Al Pacino', 'Brad Pitt']
['Al Pacino', 'Russell Crowe']
['Al Pacino', 'Denzel Washington']
['Al Pacino', 'Kevin Spacey']
['Al Pacino', 'Robert De Niro']
['Johnny Depp', 'Brad Pitt']
['Johnny Depp', 'Russell Crowe']
['Johnny Depp', 'Denzel Washington']
['Johnny Depp', 'Kevin Spacey']
['Johnny Depp', 'Robert De Niro']
['Johnny Depp', 'Al Pacino']
```