ADVANCE PYTHON ASSIGNMENT #4

IMPLEMENTATION OF TOWER OF HANOI USING LOOPS

NAME: HARIS JAMAL KAHN

CMS ID 362535

MSEE- (AI & AS) 2021 SEECS

```
import sys
# A structure to represent a stack
class Stack:
    # Constructor to set the data of
    # the newly created tree node
    def _init_(self, capacity):
        self.capacity = capacity
        self.top = -1
        self.array = [0]*capacity

# function to create a stack of given capacity.
def createStack(capacity):
    stack = Stack(capacity)
    return stack

# Stack is full when top is equal to the last index
def isFull(stack):
    return (stack.top == (stack.capacity - 1))

# Stack is empty when top is equal to -1
```

```
def isEmpty(stack):
    return (stack.top == -1)

def push(stack, item):
    if(isFull(stack)):
        return
    stack.top+=1
    stack.array[stack.top] = item

# Function to remove an item from stack.
# It decreases top by 1
def Pop(stack):
    if(isEmpty(stack)):
        return -sys.maxsize
    Top = stack.top
    stack.top-=1
    return stack.array[Top]

# Function to implement legal
# movement between two poles
def moveDisksBetweenTwoPoles(src, dest, s, d):
    pole1TopDisk = Pop(src)
    pole2TopDisk = Pop(dest)

    # When pole 1 is empty
    if (pole1TopDisk == -sys.maxsize):
        push(src, pole2TopDisk)
        moveDisk(d, s, pole2TopDisk)

    # When pole2 pole is empty
    elif (pole2TopDisk == -sys.maxsize):
        push(dest, pole1TopDisk)
```

```
        moveDisk(s, d, pole1TopDisk)

    # When top disk of pole1 > top disk of pole2
    elif (pole1TopDisk > pole2TopDisk):
        push(src, pole1TopDisk)
        push(src, pole2TopDisk)
        moveDisk(d, s, pole2TopDisk)

    # When top disk of pole1 < top disk of pole2
    else:
        push(dest, pole2TopDisk)
        push(dest, pole1TopDisk)
        moveDisk(s, d, pole1TopDisk)

# Function to show the movement of disks
def moveDisk(fromPeg, toPeg, disk):
    print("Move the disk", disk, "from '", fromPeg, "' to '", toPeg, "'")

# Function to implement TOH puzzle
def tohIterative(num_of_disks, src, aux, dest):
    s, d, a = 'S', 'D', 'A'

    # If number of disks is even, then interchange
    # destination pole and auxiliary pole
    if (num_of_disks % 2 == 0):
        temp = d
        d = a
        a = temp
    total_num_of_moves = int(pow(2, num_of_disks) - 1)

    # Larger disks will be pushed first
    for i in range(num_of_disks, 0, -1):
```

```
        push(src, i)

    for i in range(1, total_num_of_moves + 1):
        if (i % 3 == 1):
            moveDisksBetweenTwoPoles(src, dest, s, d)

        elif (i % 3 == 2):
            moveDisksBetweenTwoPoles(src, aux, s, a)

        elif (i % 3 == 0):
            moveDisksBetweenTwoPoles(aux, dest, a, d)

# Input: number of disks
num_of_disks = 3

# Create three stacks of size 'num_of_disks'
# to hold the disks
src = createStack(num_of_disks)
dest = createStack(num_of_disks)
aux = createStack(num_of_disks)

tohIterative(num_of_disks, src, aux, dest)
```