LetsUpgrade Data Structure and Algorithm Essentials
Assignment 6 | 23rd January 2021

Question 1

A Barua number is a number that consists of only zeroes and ones and has only one 1. Barua's number will start with 1. Given numbers, find out the multiplication of the numbers. Note: The input may contain one decimal number and all other Barua numbers. (Assume that each number is the very large and the total number of values give is also very large)

Input 1: 100 10 12 1000
Output 1: 12000000
Input 2: 100 121 10000000000000000
Output 2: 121000000000000000000
Input 3: 10 100 1000
Output 3: 1000000

Answer:

```
def two_factor( n ):
    twocount = 0
    while n % 2 == 0:
        twocount+=1
        n =int( n / 2)
    return twocount
def five_factor( n ):
    fivecount = 0
    while n % 5 == 0:
        fivecount+=1
        n = int(n / 5)
    return fivecount
def find_con_zero( arr, n ):
    twocount = 0
    fivecount = 0
    for i in range(n):
        twocount += two_factor(arr[i])
        fivecount += five_factor(arr[i])
```

```
    if twocount < fivecount:
        return twocount
    else:
        return fivecount
arr = [100, 10, 12, 1000]
n = 4
x = 1
N=find_con_zero(arr, n)
for i in range(n):
    if arr[i] % 10!=0:
        dec = arr[i]
        break
    else:
        dec = 1
number_str = str(dec)
x= N + len(number_str)
res = number_str.ljust(x, '0')
print(res)
```

Output:

12000000

Question 2

Implement push, pop and find the minimum element in a stack in O(1) time complexity.

Answer:

```
class Node:
    def __init__(self, value):
        self.value = value
        self.next = None
    def __str__(self):
        return "Node({})".format(self.value)
    __repr__ = __str__
```

```
class Stack:
    def __init__(self):
        self.top = None
        self.count = 0
        self.minimum = None
    def __str__(self):
        temp=self.top
        m = self.minimum
        out=[]
        if temp is None:
            print("Empty Stack")
        else:
            while not temp is None :
                val = temp.value
                if val < m:
                    m = (2 * m) -val
                    val = ( val + m ) / 2
                out.append(str(int(val)))
                temp=temp.next
            out=' '.join(out)
            return (out)
    __repr__=__str__
    def push(self,value):
        if self.top is None:
            self.top = Node(value)
            self.minimum = value
        else:
            new_node = Node(value)
            if value < self.minimum:
                temp = (2 * value) - self.minimum
                new_node.value = temp
                self.minimum = value
            new_node.next = self.top
            self.top = new_node
    def pop(self):
        new_node = self.top
```

```
        if self.top is None:
            print( "Stack is empty")
        else:
            removedNode = new_node.value
            if removedNode < self.minimum:
                self.minimum = ( ( 2 * self.minimum ) - removedNode )
                new_node.value = ( (removedNode + self.minimum) / 2 )
            self.top = self.top.next
            return int(new_node.value)
    def getMin(self):
        if self.top is None:
            return "Stack is empty"
        else:
            return self.minimum
stack = Stack()
stack.push(5)
stack.push(4)
stack.push(3)
stack.push(2)
print("Initial stack:", stack)
print("Popped Element:", stack.pop())
print("New Stack:", stack)
print("Minimum element:", stack.getMin())
```

Output:

Initial stack: 2 3 4 5

Popped Element: 2

New Stack: 3 4 5

Minimum element: 3

Submitted To: Subrat Kumar Swain *(LetsUpgrade Instructor)*

Submitted By: Arun Sharma

curiousarun08@gmail.com