**Practice Problems in Data Structures**

BStat, 1st Year 2023–2024

**Date**: 06 October, 2023

| **PRACTICING INSTRUCTIONS** |
| --- |
| 1. Write down the best possible implementation of the problems listed below. |
| 2. Take care of the error cases too. |
| 3. After you finish the coding, look at the model answers at the end and understand. |

**NOTE:** The programs are to be written in Python and should be well commented. All programs should take the required inputs from standard input buffer and print the desired outputs to the standard output buffer, until otherwise stated.

Q1. (EASY) Given an expression of parentheses in the form of string, write a program using stack to examine whether the expression is `valid` or not. The validity refers to appropriate matching of the pairs of parentheses in the given expression.

Q2. (EASY) Given a set of positive integers, print the next greater integer for every element in the set. The elements for which no greater element exist, print the next greater integer as -1.

**Model Answers:**

Q1. GENERAL APPROACH

```
def CheckValidity(expression):
    StackDS = []
    for c in expression:
        if c in ['(', '{', '[']:
            StackDS.append(c)
        else:
            if not StackDS:
                return False
            current_c = StackDS.pop()
            if current_c == '(':
                if c != ')':
                    return False
            if current_c == '{':
                if c != '}':
                    return False
            if current_c == '[':
                if c != ']':
                    return False
    if StackDS:
        return False
    return True
expression = input('Enter an expression: ')
if CheckValidity(expression):
    print('VALID')
else:
    print('INVALID')
```

BETTER APPROACH

...

Q2. GENERAL APPROACH

```
def isEmpty(stack):
    return len(stack) == 0
def push(stack, x):
    stack.append(x)
def pop(stack):
    if isEmpty(stack):
        print("Error : stack underflow")
    else:
        return stack.pop()
def printStack(arr):
```

```
    StackDS = []
    element = 0
    next = 0
    push(StackDS, arr[0])
    for i in range(1, len(arr), 1):
        next = arr[i]
        if isEmpty(StackDS) == False:
            element = pop(StackDS)
            while element < next:
                print(str(element) + " -> " + str(next))
                if isEmpty(StackDS) == True:
                    break
                element = pop(StackDS)
            if element > next:
                push(StackDS, element)
        push(StackDS, next)
    while isEmpty(StackDS) == False:
        element = pop(StackDS)
        next = -1
        print(str(element) + " -> " + str(next))
s = input('Enter a set of positive integers: ').split()
printStack(s)
```

BETTER APPROACH

...