Q.Given an array **arr[ ]** of size **N** having elements, the task is to find the next greater element for each element of the array in order of their appearance in the array.Next greater element of an element in the array is the nearest element on the right which is greater than the current element.If there does not exist next greater of current element, then next greater element for current element is -1. For example, next greater of the last element is always -1.

Input:

N = 4, arr[] = [1 3 2 4]

Output:

3 4 4 -1

Explanation:

In the array, the next larger element

to 1 is 3 , 3 is 4 , 2 is 4 and for 4 ?

since it doesn't exist, it is -1.

Input:

N = 5, arr[] [6 8 0 1 3]

Output:

8 -1 1 3 -1

Explanation:

In the array, the next larger element to

6 is 8, for 8 there is no larger elements

hence it is -1, for 0 it is 1 , for 1 it

is 3 and then for 3 there is no larger

element on right and hence -1.

Q. Given an array **a**of integers of length **n**, find the nearest smaller number for every element such that the smaller element is on left side.If no small element present on the left print -1.

Input: n = 3

a = {1, 6, 2}

Output: -1 1 1

Explaination: There is no number at the

left of 1. Smaller number than 6 and 2 is 1.

Input: n = 6

a = {1, 5, 0, 3, 4, 5}

Output: -1 1 -1 0 3 4

Explaination: Upto 3 it is easy to see

the smaller numbers. But for 4 the smaller

numbers are 1, 0 and 3. But among them 3

is closest. Similary for 5 it is 4.

Q. Implement a Stack using two queues**q1** and**q2**.

Input:

push(2)

push(3)

pop()

push(4)

pop()

Output:3 4

Explanation:

push(2) the stack will be {2}

push(3) the stack will be {2 3}

pop() poped element will be 3 the

  stack will be {2}

push(4) the stack will be {2 4}

pop()   poped element will be 4

Input:

push(2)

pop()

pop()

push(3)

Output:2 -1

Q. You are given a stack **St**. You have to reverse the stack using recursion.

Input:St = {3,2,1,7,6}

Output:{6,7,1,2,3}

Input:St = {4,3,9,6}

Output:{6,9,3,4}

Q. You are given a string **S**, the task is to reverse the string using stack.

Input: S="GeeksforGeeks"

Output: skeeGrofskeeG

Q. Given string **S** representing a postfix expression, the task is to evaluate the expression and find the final value. Operators will only include the basic arithmetic operators like \***, /, + and -**.

**Example 1:**

Input: S = "231\*+9-"

Output: -4

Explanation:

After solving the given expression,

we have -4 as result.

Input: S = "123+\*8-"

Output: -3

Explanation:

After solving the given postfix

expression, we have -3 as result.

Q. Design a stack that supports push, pop, top, and retrieving the minimum element in constant time.

Implement the MinStack class:

* MinStack() initializes the stack object.
* void push(int val) pushes the element val onto the stack.
* void pop() removes the element on the top of the stack.
* int top() gets the top element of the stack.
* int getMin() retrieves the minimum element in the stack.

You must implement a solution with O(1) time complexity for each function.

**Example 1:**

Input

["MinStack","push","push","push","getMin","pop","top","getMin"]

[[],[-2],[0],[-3],[],[],[],[]]

Output

[null,null,null,null,-3,null,0,-2]

Explanation

MinStack minStack = new MinStack();

minStack.push(-2);

minStack.push(0);

minStack.push(-3);

minStack.getMin(); // return -3

minStack.pop();

minStack.top(); // return 0

minStack.getMin(); // return -2

Ans: Here are the solutions to the given problems:

1. Next Greater Element:

```python

def nextGreaterElement(arr):

stack = []

result = []

for i in range(len(arr)-1, -1, -1):

while stack and stack[-1] <= arr[i]:

stack.pop()

if not stack:

result.append(-1)

else:

result.append(stack[-1])

stack.append(arr[i])

result.reverse()

return result

# Example usage:

arr = [1, 3, 2, 4]

print(nextGreaterElement(arr)) # Output: [3, 4, 4, -1]

```

2. Nearest Smaller Number:

```python

def nearestSmallerNumber(a):

stack = []

result = []

for num in a:

while stack and stack[-1] >= num:

stack.pop()

if not stack:

result.append(-1)

else:

result.append(stack[-1])

stack.append(num)

return result

# Example usage:

a = [1, 6, 2]

print(nearestSmallerNumber(a)) # Output: [-1, 1, 1]

```

3. Stack using Queues:

```python

from queue import Queue

class Stack:

def \_\_init\_\_(self):

self.q1 = Queue()

self.q2 = Queue()

def push(self, x):

self.q2.put(x)

while not self.q1.empty():

self.q2.put(self.q1.get())

self.q1, self.q2 = self.q2, self.q1

def pop(self):

if not self.q1.empty():

self.q1.get()

def top(self):

if not self.q1.empty():

return self.q1.queue[0]

def empty(self):

return self.q1.empty()

# Example usage:

stack = Stack()

stack.push(2)

stack.push(3)

stack.pop()

stack.push(4)

stack.pop()

print(stack.top()) # Output: 3

```

4. Reverse Stack using Recursion:

```python

def reverseStack(stack):

if not stack:

return

temp = stack.pop()

reverseStack(stack)

insertAtBottom(stack, temp)

def insertAtBottom(stack, item):

if not stack:

stack.append(item)

return

temp = stack.pop()

insertAtBottom(stack, item)

stack.append(temp)

# Example usage:

stack = [3, 2, 1, 7, 6]

reverseStack(stack)

print(stack) # Output: [6, 7, 1, 2, 3]

```

5. Reverse String using Stack:

```python

def reverseString(s):

stack = []

for char in s:

stack.append(char)

reversed\_str = ""

while stack:

reversed\_str += stack.pop()

return reversed\_str

# Example usage:

s = "GeeksforGeeks"

print(reverseString(s)) # Output: skeeGrofskeeG

```

6. Evaluate Postfix Expression:

```python

def evaluatePostfix(S):

stack = []

for char in S:

if char.isdigit():

stack.append(int(char))

else:

operand2 = stack.pop()

operand1 = stack.pop()

result = evaluateExpression(char, operand1, operand2)

stack.append(result)

return stack.pop()

def evaluateExpression(operator, operand1, operand2):

if operator == '\*':

return operand1 \* operand2

elif operator == '/':

return operand1 / operand2

elif operator == '+':

return operand1 + operand2

elif operator == '-':

return operand1 - operand2

# Example usage:

S = "231\*+9-"

print(evaluatePostfix(S)) # Output: -4

```

7. MinStack:

```python

class MinStack:

def \_\_init\_\_(self):

self.stack = []

self.min\_stack = []

def push(self, val):

self.stack.append(val)

if not self.min\_stack or val <= self.min\_stack[-1]:

self.min\_stack.append(val)

def pop(self):

if self.stack:

if self.stack[-1] == self.min\_stack[-1]:

self.min\_stack.pop()

self.stack.pop()

def top(self):

if self.stack:

return self.stack[-1]

def getMin(self):

if self.min\_stack:

return self.min\_stack[-1]

# Example usage:

minStack = MinStack()

minStack.push(-2)

minStack.push(0)

minStack.push(-3)

print(minStack.getMin()) # Output: -3

minStack.pop()

print(minStack.top()) # Output: 0

print(minStack.getMin()) # Output: -2

```

These are the solutions to the given problems. Let me know if you need any further assistance!