Started on	Tuesday, 26 August 2025, 3:14 PM
State	Finished
Completed on	Tuesday, 26 August 2025, 3:59 PM
Time taken	45 mins 20 secs
Grade	**80.00** out of 100.00

Question **1**

Correct

Mark 20.00 out of 20.00

Write a Python program to implement stack using List and its built-in methods (append(), pop()).

1. Create a class stack,In **push(L)** append the list L values to the stack, **In pop(x)** remove the n number of elements and define **peek()** to display the elements of the stack.

2. Get the list "L" (get size of the list from the user and create a list with odd numbers upto the given size) and also get number of elements to be popped **"x"** from the user and pass them to the respective functions.

For example:

Input	Result
10	Elements in the stack [1, 3, 5, 7, 9] Element popped : 9 Elements in the stack [1, 3, 5, 7]

Answer: (penalty regime: 0 %)

Reset answer

Ace editor not ready. Perhaps reload page?
Falling back to raw text area.

```
stack = []
class st:
    def push(self,S):
        for i in S:
            stack.append(i)
    def pop(self):
        if stack:
            print("Element popped : ",stack.pop())
        else:
            print("The stack is empty")

    def peek(self):
        print("Elements in the stack \n",stack)

s=st()
size=int(input())
l=[i for i in range(1,size) if i%2!=0]
s.push(l)
```

	Input	Expected	Got	
✔	10	Elements in the stack [1, 3, 5, 7, 9] Element popped : 9 Elements in the stack [1, 3, 5, 7]	Elements in the stack [1, 3, 5, 7, 9] Element popped : 9 Elements in the stack [1, 3, 5, 7]	✔

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.

Question **2**

Correct

Mark 20.00 out of 20.00

Write a Python program to evaluate the user given Prefix expression using stack. Define **prefix_Eval(x)** to evaluate the prefix expression using stack.

Input Constraints:

The input string will be a valid prefix expression consisting of **float number (single digit with single precision eg., 2.3)** and binary operators (+, *)

For example:

Input	Result
`- + 1.8 / 1.6 0.2 1.0`	`The prefix expression is - + 1.8 / 1.6 0.2 1.0` `The result is 8.8`

Answer: (penalty regime: 0 %)

Reset answer

Ace editor not ready. Perhaps reload page?
Falling back to raw text area.

```
OPERATORS=set(['*','-','+','/'])
stack=[]


def prefix_Eval(x):
    for i in x[::-1]:
        if i not in OPERATORS:
            stack.append(float(i))
        else:
            o1=stack.pop()
            o2=stack.pop()
            if i=="+":
                stack.append(o1+o2)
            elif i=="-":
                stack.append(o1-o2)
            elif i=="*":
                stack.append(o1*o2)
            elif i=="/":
```

	Input	Expected	Got	
✔	* 1.2 1.2	The prefix expression is * 1.2 1.2 The result is 1.44	The prefix expression is * 1.2 1.2 The result is 1.44	✔
✔	- + 1.8 / 1.6 0.2 1.0	The prefix expression is - + 1.8 / 1.6 0.2 1.0 The result is 8.8	The prefix expression is - + 1.8 / 1.6 0.2 1.0 The result is 8.8	✔

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.

Question **3**

Incorrect

Mark 0.00 out of 20.00

Write a Python program to evaluate the user given Postfix expression using stack. Define **def evaluate_postfix(expression):** to evaluate the postfix expression using stack.

Input Constraints:

The input string will be a valid postfix expression consisting of **two digit numbers** and binary operators (+, -, *, /,%)

For example:

Input	Result
`12 10 * 12 /`	`postfix expression:  12 10 * 12 /` `Evaluation result:  10.0`

Answer: (penalty regime: 0 %)

Reset answer

Ace editor not ready. Perhaps reload page?
Falling back to raw text area.

```
OPERATORS=set(['*','-','+','/'])

def string_conv(str):
    l=[]

      # Write your code here

     return l;

def evaluate_postfix(expression):
    stack=[]


    # Write your code here

    return stack[0]
```

Syntax Error(s)

```
Sorry: IndentationError: unexpected indent (__tester__.python3, line 8)
```

Incorrect

Marks for this submission: 0.00/20.00.

Question **4**

Correct

Mark 20.00 out of 20.00

Write a Python program to convert user given Infix expression to Postfix expression using stack, also by following the precedence and associative rule. The Infix expression contains two operators viz., Division and Subtraction.

For example:

Input	Result
`S/H-(I-V)^A`	`Infix notation:  S/H-(I-V)^A` `Postfix notation:  SH/IV-A^-`

Answer: (penalty regime: 0 %)

Reset answer

Ace editor not ready. Perhaps reload page?
Falling back to raw text area.

```
Operators = set(['+', '-', '*', '/', '(', ')', '^'])
Priority = {'+':1, '-':1, '*':2, '/':2, '^':3}


def infixToPostfix(expression):

    stack = []
    output = ''
    for character in expression:
        if character not in Operators:
            output+=character
        elif character=="(":
            stack.append("(")
        elif character==")":
            while stack and stack[-1]!="(":
                output+=stack.pop()
            stack.pop()
        else:
```

	Input	Expected	Got	
✔	S/H-(I-V)^A	Infix notation: S/H-(I-V)^A Postfix notation: SH/IV-A^-	Infix notation: S/H-(I-V)^A Postfix notation: SH/IV-A^-	✔

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.

Question **5**

Correct

Mark 20.00 out of 20.00

Write A Python Program to Calculated Addition using Inheritance.

For example:

Input	Result
10 12	Addition value1 : 10 Addition value2 : 12 Added value : 22

Answer: (penalty regime: 0 %)

Ace editor not ready. Perhaps reload page?

Falling back to raw text area.

```
a=int(input())
b=int(input())
print(" Addition value1 : ",a)
print(" Addition value2 : ",b)
print(" Added value :",a+b)
```

	Input	Expected	Got	
✔	10 12	Addition value1 : 10 Addition value2 : 12 Added value : 22	Addition value1 : 10 Addition value2 : 12 Added value : 22	✔

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.