Week 3 Programming Assignment

Due on 2020-02-20, 23:59 IST

Write three Python functions as specified below. Paste the text for all three functions together into the submission window.

- You may define additional auxiliary functions as needed.
- In all cases you may assume that the value passed to the function is of the expected type, so your function does not have to check for malformed inputs.
- For each function, there are some public test cases and some (hidden) private test cases.
- "Compile and run" will evaluate your submission against the public test cases.
- "Submit" will evaluate your submission against the hidden private test cases and report a score on 100. There are 10 private testcases in all, each with equal weightage.
- Ignore warnings about "Presentation errors".

---

1. Define a Python function `remdup(l)` that takes a nonempty list of integers `l` and removes all duplicates in `l`, keeping only the first occurrence of each number. For instance:

```
>>> remdup([3,1,3,5])
[3, 1, 5]

>>> remdup([7,3,-1,-5])
[7, 3, -1, -5]

>>> remdup([3,5,7,5,3,7,10])
[3, 5, 7, 10]
```

2. Write a Python function [illegible] that takes a nonempty list of

2. Write a Python function `sumsquare(l)` that takes a nonempty list of integers and returns a list `[odd,even]`, where `odd` is the sum of squares all the odd numbers in `l` and `even` is the sum of squares of all the even numbers in `l`.

   Here are some examples to show how your function should work.

```
>>> sumsquare([1,3,5])
[35, 0]

>>> sumsquare([2,4,6])
[0, 56]

>>> sumsquare([-1,-2,3,7])
[59, 4]
```

3. A two dimensional matrix can be represented in Python row-wise, as a list of lists: each inner list represents one row of the matrix. For instance, the matrix

```
1  2  3  4
5  6  7  8
```

   would be represented as `[[1, 2, 3, 4], [5, 6, 7, 8]]`.

   The transpose of a matrix converts each row into a column. The transpose of the matrix above is:

```
1  5
2  6
3  7
4  8
```

   which would be represented as `[[1, 5], [2, 6], [3, 7], [4, 8]]`.

   Write a Python function `transpose(m)` that takes as input a two dimensional matrix `m` and returns the transpose of `m`. **The argument `m` should remain undisturbed by the function**.

   Here are some examples to show how your function should work. You may assume that the input to the function is always a non-empty matrix.

```
>>> transpose([[1,2,3],[4,5,6]])
[[1, 4], [2, 5], [3, 6]]

>>> transpose([[1],[2],[3]])
[[1, 2, 3]]

>>> transpose([[3]])
[[3]]
```

Sample Test Cases

	Input	Output
Test Case 1	`remdup([5,5,5,5,1,1,5,5,5])`	`[5, 1]`
Test Case 2	`remdup([8,6,4,6,8])`	`[8, 6, 4]`
Test Case 3	`remdup([5])`	`[5]`
Test Case 4	`remdup([])`	`[]`
Test Case 5	`sumsquare([1,2,3,4,5,6])`	`[35, 56]`
Test Case 6	`sumsquare([1,4,9,16,25,36,49,64])`	`[3108, 5664]`
Test Case 7	`sumsquare([0,1,-1,0,2,-2,3,-3])`	`[20, 8]`
Test Case 8	`transpose([[1,2,3],[4,5,6],[7,8,9]])`	`[[1, 4, 7], [2, 5, 8], [3, 6, 9]]`
Test Case 9	`transpose([[1,2,3,4]])`	`[[1], [2], [3], [4]]`
Test Case 10	`transpose([[1,0,0],[0,1,0],[0,0,1]])`	`[[1, 0, 0], [0, 1, 0], [0, 0, 1]]`
Test Case 11	`remdup([3,1,3,5])`	`[3, 1, 5]`
Test Case 12	`remdup([7,3,-1,-5])`	`[7, 3, -1, -5]`
Test Case 13	`remdup([3,5,7,5,3,7,10])`	`[3, 5, 7, 10]`
Test Case 14	`sumsquare([1,3,5])`	`[35, 0]`
Test Case 15	`sumsquare([2,4,6])`	`[0, 56]`
Test Case 16	`sumsquare([-1,-2,3,7])`	`[59, 4]`
Test Case 17	`transpose([[1,2,3],[4,5,6]])`	`[[1, 4], [2, 5], [3, 6]]`
Test Case 18	`transpose([[1],[2],[3]])`	`[[1, 2, 3]]`
Test Case 19	`transpose([[3]])`	`[[3]]`

The due date for submitting this assignment has passed.

As per our records you have not submitted this assignment.

Sample solutions (Provided by instructor)

```
def remdup(l):
    return(myremdup(l,[]))
```

Mar 2021, 10:00-12:00 ()

Online Programming Test 2, 09 Mar 2021, 20:00-22:00 ()

```
3
4  def myremdup(l,s):
5      if l == []:
6          return([])
7      else:
8          if l[0] in s:
9              return(myremdup(l[1:],s))
10         else:
11             return([l[0]]+myremdup(l[1:],s+[l[0]]))
12
13 ####################
14
15 def even(n):
16     return(n%2 == 0)
17
18 def sumsquare(l):
19     oddsum = 0
20     evensum = 0
21     for n in l:
22         if even(n):
23             evensum += n*n
24         else:
25             oddsum += n*n
26     return([oddsum,evensum])
27
28 ###################
29
30 def transpose(l):
31   outl = []
32   for row in l[:1]:
33     for i in range(len(row)):
34       outl.append([])
35   for row in l:
36     for i in range(len(row)):
37       outl[i].append(row[i])
38   return(outl)
39 import ast
40
41 def parse(inp):
42   inp = ast.literal_eval(inp)
43   return (inp)
44
45 fncall = input()
46 lparen = fncall.find("(")
47 rparen = fncall.rfind(")")
48 fname = fncall[:lparen]
49 farg = fncall[lparen+1:rparen]
50
51 if fname == "remdup":
52   arg = parse(farg)
53   print(remdup(arg))
54
55 if fname == "sumsquare":
56   arg = parse(farg)
57   print(sumsquare(arg))
58
59 if fname == "transpose":
60   arg = parse(farg)
61   savearg = arg
62   ans = transpose(arg)
63   if savearg == arg:
64     print(ans)
65   else:
66     print("Side effect")
67
68
```