

Started on Tuesday, 7 May 2024, 6:24 PM

State Finished

Completed on Tuesday, 7 May 2024, 6:35 PM

Time taken 10 mins 32 secs

Marks 5.00/5.00

Grade **50.00** out of 50.00 (**100%**)

Name [DWIJESH SREERAM S 2022-CSD-A](#)



Question 1

Correct

Mark 1.00 out of 1.00

A string with parentheses is well bracketed if all parentheses are matched: every opening bracket has a matching closing bracket and vice versa.

Write a Python function `wellbracketed(s)` that takes a string `s` containing parentheses and returns `True` if `s` is well bracketed and `False` otherwise.

Hint: Keep track of the nesting depth of brackets. Initially the depth is 0. The depth increases with each opening bracket and decreases with each closing bracket. What are the constraints on the value of the nesting depth for the string to be wellbracketed?

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

```
>>> wellbracketed("22")
False
```

```
>>> wellbracketed("(a+b)(a-b)")
True
```

```
>>> wellbracketed("(a(b+c)-d)((e+f)")
False
```

Answer: (penalty regime: 0 %)

Reset answer

```
1 def wellbracketed(s):
2     stack = []
3     for c in s:
4         if c == '(':
5             stack.append(c)
6         elif c == ')':
7             if not stack or stack.pop() != '(':
8                 return False
9     return not stack
10
```

	Test	Expected	Got	
✓	<code>print(wellbracketed("22"))</code>	False	False	✓
✓	<code>print(wellbracketed("(a+b)(a-b)"))</code>	True	True	✓
✓	<code>print(wellbracketed("(a(b+c)-d)((e+f)"))</code>	False	False	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.



Question 2

Correct

Mark 1.00 out of 1.00

In this exercise you will write a function that determines whether or not a password is good. We will define a good password to be a one that is at least 8 characters long and contains at least one uppercase letter, at least one lowercase letter, and at least one number. Your function should return True if the password passed to it as its only parameter is good. Otherwise it should return False. Include a main program that reads a password from the user and reports whether or not it is good. Ensure that your main program only runs when your solution has not been imported into another file.

Sample Input 1

chennai

Sample Output 1

That isn't a good password.

Sample Input 2

Chennai18

Sample Output 2

That's a good password.

Answer: (penalty regime: 0 %)

Reset answer

```
1 def checkPassword(input1):
2     if len(input1)<8:
3         print("That isn't a good password.")
4         return
5
6     has_upper=False
7     has_lower=False
8     has_digit=False
9
10    for char in input1:
11        if char.isupper():
12            has_upper=True
13        elif char.islower():
14            has_lower=True
15        elif char.isdigit():
16            has_digit=True
17
18    if has_upper and has_lower and has_digit:
19        print("That's a good password.")
20    else:
21        print("That isn't a good password.")
```

	Test	Expected	Got	
✓	checkPassword('chennai')	That isn't a good password.	That isn't a good password.	✓
✓	checkPassword('Chennai18')	That's a good password.	That's a good password.	✓

Passed all tests! ✓

Correct



Marks for this submission: 1.00/1.00.

Question **3**

Correct

Mark 1.00 out of 1.00

Write a Python function `sumofsquares(m)` that takes an integer `m` returns `True` if `m` is a sum of squares and `False` otherwise. (If `m` is not positive, your function should return `False`.)

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

```
>>> sumofsquares(41)
```

```
True
```

```
>>> sumofsquares(30)
```

```
False
```

```
>>> sumofsquares(17)
```

```
True
```

Answer: (penalty regime: 0 %)

Reset answer

```
1 from math import *
2
3 def issquare(n):
4     k = int(sqrt(n))
5     return(k*k == n)
6
7 def sumofsquares(m):
8     if m<=0:
9         return False
10    i=0
11    while i**2<=m:
12        sqr =m-i**2
13        j= int(sqr**0.5)
14        if j**2 == sqr:
15            return True
16        i+=1
17    return False
```

	Test	Expected	Got	
✓	<code>print(sumofsquares(41))</code>	True	True	✓
✓	<code>print(sumofsquares(30))</code>	False	False	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.



Question 4

Correct

Mark 1.00 out of 1.00

Write a program that reads values from the user until a blank line is entered. Display the total of all of the values entered by the user (or 0 if the first value entered is a blank line). Complete this task using recursion. Your program may not use any loops.

Hint: The body of your recursive function will need to read one value from the user, and then determine whether or not to make a recursive call. Your function does not need to take any arguments, but it will need to return a numeric result.

Sample Input

5
10
15
20
25

Sample Output

75

Answer: (penalty regime: 0 %)

Reset answer

```
1 def readAndTotal():
2     value = input("")
3     if not value:
4         return 0
5     else:
6         try:
7             # Convert input to float for handling decimals
8             num = float(value)
9             return num + readAndTotal() # Recursive call with sum
10    except ValueError:
11        print("Invalid input. Please enter a number.")
12        return get_total() # Retry on invalid input
13
14 # Get the total from the user
15 total = readAndTotal()
16 print("%.0f"%total)
17
18
```

	Input	Expected	Got	
✓	5 10 15 20 25	75	75	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.



Question 5

Correct

Mark 1.00 out of 1.00

A list rotation consists of taking the last element and moving it to the front. For instance, if we rotate the list [1,2,3,4,5], we get [5,1,2,3,4]. If we rotate it again, we get [4,5,1,2,3].

Write a Python function `rotatelist(l,k)` that takes a list `l` and a positive integer `k` and returns the list `l` after `k` rotations. If `k` is not positive, your function should return `l` unchanged. Note that your function should not change `l` itself, and should return the rotated list.

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

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

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

```
>>> rotatelist([1,2,3,4,5],12)
[4, 5, 1, 2, 3]
```

Answer: (penalty regime: 0 %)

Reset answer

```
1 def rotatelist(l, k):
2     # Check if k is positive
3     if k <= 0:
4         return l
5
6     # Calculate the effective rotation by taking modulo with the length of the list
7     k = k % len(l)
8
9     # Rotate the list by moving the last k elements to the front
10    rotated_list = l[-k:] + l[:-k]
11
12    return rotated_list
```

	Test	Expected	Got	
✓	<code>print(rotatelist([1,2,3,4,5],1))</code>	[5, 1, 2, 3, 4]	[5, 1, 2, 3, 4]	✓
✓	<code>print(rotatelist([1,2,3,4,5],3))</code>	[3, 4, 5, 1, 2]	[3, 4, 5, 1, 2]	✓
✓	<code>print(rotatelist([1,2,3,4,5],12))</code>	[4, 5, 1, 2, 3]	[4, 5, 1, 2, 3]	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.



[◀ Week-07_MCQ](#)

Jump to...

[WEEK-07-Extra ▶](#)

