Started on	Friday, 17 May 2024, 12:06 PM
State	Finished
Completed on	Friday, 17 May 2024, 2:02 PM
Time taken	1 hour 56 mins
Marks	5.00/5.00
Grade	**50.00** out of 50.00 (**100**%)
Name	JUNIDE CHRIS A 2022-CSD-A

Question **1**

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

```
def checkPassword(input1):
    if len(input1) < 8:
        print("That isn't a good password.")
        return

    has_upper = False
    has_lower = False
    has_digit = False

    for char in input1:
        if char.isupper():
            has_upper = True
        elif char.islower():
            has_lower = True
        elif char.isdigit():
            has_digit = True

    if has_upper and has_lower and has_digit:
        print("That's a good password.")
    else:
        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 **2**

Correct

Mark 1.00 out of 1.00

Euclid was a Greek mathematician who lived approximately 2,300 years ago. His algorithm for computing the greatest common divisor of two positive integers, a and b, is both efficient and recursive. It is outlined below:

If b is 0 then
  eturn a
Else
  Set c equal to the remainder when a is divided by b
  Return the greatest common divisor of b and c

Write a program that implements Euclid's algorithm and uses it to determine the greatest common divisor of two integers entered by the user. Test your program with some very large integers. The result will be computed quickly, even for huge numbers consisting of hundreds of digits, because Euclid's algorithm is extremely efficient.

Answer: (penalty regime: 0 %)

```
def gcd(a,b):
    if b==0:
        return a
    else:
        c=a%b
        return gcd(b,c)
a=int(input())
b=int(input())
x=gcd(a,b)
print(x)
```

	Input	Expected	Got	
✔	8 12	4	4	✔
✔	720 1000	40	40	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Question **3**

Correct

Mark 1.00 out of 1.00

The notion of a palindrome was introduced previously. In this exercise you will write a recursive function that determines whether or not a string is a palindrome. The empty string is a palindrome, as is any string containing only one character. Any longer string is a palindrome if its first and last characters match, and if the string formed by removing the first and last characters is also a palindrome.

Write a program that reads a string from the user and uses your recursive function to determine whether or not it is a palindrome. Then your program should display an appropriate message for the user.

Sample Input

malayalam

Sample Output

That was a palindrome!

Sample Input

madan

Sample Output

That is not a palindrome.

Answer: (penalty regime: 0 %)

Reset answer

```
def isPalindrome(s):
    s1=''.join(reversed(s))
    if s1==s:
         return True
    return False
line=input()
if isPalindrome(line):
     print("That was a palindrome!")
else:
     print("That is not a palindrome.")
```

	Input	Expected	Got	
✔	malayalam	That was a palindrome!	That was a palindrome!	✔
✔	madan	That is not a palindrome.	That is not a palindrome.	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Question **4**

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

```
def rotatelist(l, k):
    if k <= 0:
        return l

    k = k % len(l)
    rotated_list = []

    for i in range(len(l)):
        rotated_list.append(l[(i + len(l) - k) % len(l)])

    return rotated_list
```

	Test	Expected	Got	
✔	print(rotatelist([1,2,3,4,5],1))	[5, 1, 2, 3, 4]	[5, 1, 2, 3, 4]	✔
✔	print(rotatelist([1,2,3,4,5],3))	[3, 4, 5, 1, 2]	[3, 4, 5, 1, 2]	✔
✔	print(rotatelist([1,2,3,4,5],12))	[4, 5, 1, 2, 3]	[4, 5, 1, 2, 3]	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Question **5**

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

```
def wellbracketed(s):
    l=[]
    count=0
    for i in s:
        if i=='(':
           count=count+1
        if i==')':
            count=count-1
    if count==0:
        return True
    else:
        return False
```

	Test	Expected	Got	
✔	print(wellbracketed("22)"))	False	False	✔
✔	print(wellbracketed("(a+b)(a-b)"))	True	True	✔
✔	print(wellbracketed("(a(b+c)-d)((e+f)"))	False	False	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

◄ Week-07_MCQ

Jump to...

WEEK-07-Extra ►