Started on	Monday, 15 April 2024, 9:24 PM
State	Finished
Completed on	Sunday, 21 April 2024, 4:30 PM
Time taken	5 days 19 hours
Marks	5.00/5.00
Grade	**50.00** out of 50.00 (**100**%)
Name	ABHIGNYA P 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

```
def wellbracketed(s):
    depth=0
    for char in s:
        if char=='(':
            depth+=1
        elif char==')':
            depth-=1
            if depth<0:
                return False
    return depth==0
```

	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.

Question **2**

Correct

Mark 1.00 out of 1.00

Write a function that takes three numbers as parameters, and returns the median value of those parameters as its result.

Answer: (penalty regime: 0 %)

Reset answer

```
def median(a,b,c):
    if a<b<c or c<b<a:
        return b
    elif b<a<c or c<a<b:
        return a
    else:
        return c
```

	Test	Expected	Got	
✔	`print(median(10, 20, 30))`	20	20	✔
✔	`print(median(60, 50, 40))`	50	50	✔
✔	`print(median(70, 90, 80))`	80	80	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Question **3**

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 **4**

Correct

Mark 1.00 out of 1.00

A prime number is an integer greater than one that is only divisible by one and itself. Write a function that determines whether or not its parameter is prime, returning True if it is, and False otherwise.

Answer: (penalty regime: 0 %)

Reset answer

```
def isPrime(n):
    if n<=1:
        return False
    elif n<=3:
        return True
    elif n%2==0 or n%3==0:
        return False
    i=5
    while i*i<=n:
        if n%i==0 or n%(i+2)==0:
            return False
        i+=6
    return True
```

	Test	Expected	Got	
✔	print(isPrime(1))	False	False	✔
✔	print(isPrime(2))	True	True	✔
✔	print(isPrime(3))	True	True	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Question **5**

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

```
def sum_values():
    user_input=input().strip()
    if user_input=="":
        return 0
    value=int(user_input)
    return value+sum_values()
total=sum_values()
print(total)

```

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

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

◄ Week-07_MCQ

Jump to...