Started on	Monday, 13 May 2024, 7:26 PM
State	Finished
Completed on	Monday, 13 May 2024, 8:18 PM
Time taken	51 mins 27 secs
Marks	5.00/5.00
Grade	**50.00** out of 50.00 (**100**%)
Name	AKSAYAA S V 2022-CSD-A

Question **1**

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 **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

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 **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

```
total = 0
input_count = 0

while True:
    try:
        line = input()
        if line.strip() == "":
            break
        num = int(line)
        total += num
        input_count += 1
    except ValueError:
        continue

if input_count == 0:
    print(0)
else:
    print(total)
```

	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

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.

◄ Week-07_MCQ

Jump to...

WEEK-07-Extra ►