Started on	Wednesday, 17 April 2024, 11:05 AM
State	Finished
Completed on	Monday, 22 April 2024, 12:08 PM
Time taken	5 days 1 hour
Marks	5.00/5.00
Grade	**50.00** out of 50.00 (**100**%)
Name	TAMILARASI R 2022-CSD-A

Question **1**

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

```
    # Base case: an empty string or a string with one character i
    if len(s) <= 1:
        return True
    # Check if the first and last characters match
    elif s[0] == s[-1]:
        # Recursively check if the substring formed by removing t
        return is_palindrome(s[1:-1])
    else:
        return False

# Read a string from the user
s = input()

# Check if the string is a palindrome using the is_palindrome fur
if is_palindrome(s):
    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 **2**

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):
    c=0
    for i in s:
        if i=='(':
            c+=1
        elif i==')':
            c+=1
        else:
            continue
    if c%2==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

Question **3**

Correct

Mark 1.00 out of 1.00

Given an integer n, return an list of length n + 1 such that for each i (0 <= i <= n), ans[i] is the number of 1's in the binary representation of i.

Example:

```
Input: n = 2
Output: [0,1,1]
Explanation:
0 --> 0
1 --> 1
2 --> 10
```

Example2:

```
Input: n = 5
Output: [0,1,1,2,1,2]
Explanation:
0 --> 0
1 --> 1
2 --> 10
3 --> 11
4 --> 100
5 --> 101
```

Note: Complete the given function alone

For example:

Test	Result
`print(CountingBits(5))`	`[0, 1, 1, 2, 1, 2]`

Answer: (penalty regime: 0 %)

Reset answer

```
def CountingBits(n):
    return [bin(i).count('1') for i in range(n+1)]

```

	Test	Expected	Got	
✔	`print(CountingBits(2))`	`[0, 1, 1]`	`[0, 1, 1]`	✔
✔	`print(CountingBits(5))`	`[0, 1, 1, 2, 1, 2]`	`[0, 1, 1, 2, 1, 2]`	✔

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

```
def sum_values():
    # Read a value from the user
    value = input().strip()

    # Base case: If the user enters a blank line, return 0
    if value == '':
        return 0

    # Convert the input value to an integer and add it to the sum o
    return int(value) + sum_values()

# Main program

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.

Question **5**

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:
        return gcd(b, a % b)

def main():
    # Read two integers from the user
    a = int(input())
    b = int(input())

    # Compute the GCD using Euclid's algorithm
    result = gcd(a, b)

    # Display the result
    print(result)

if __name__ == "__main__":
    main()
```

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

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

◄ Week-07_MCQ

Jump to...