
Started on Friday, 26 April 2024, 9:55 PM

State Finished

Completed on Friday, 26 April 2024, 10:37 PM

Time taken 42 mins 10 secs

Marks 5.00/5.00

Grade 50.00 out of 50.00 (100%)

Name [ISHWARYA M 2022-CSD-A](#)

Question 1

Correct

Mark 1.00 out of 1.00

Given an integer n , return an list of length $n + 1$ such that for each i ($0 \leq i \leq n$), $\text{ans}[i]$ 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
<code>print(CountingBits(5))</code>	<code>[0, 1, 1, 2, 1, 2]</code>

Answer: (penalty regime: 0 %)

Reset answer

```
1 def CountingBits(n):  
2     result = [0] * (n + 1)  
3     for i in range(n + 1):  
4         result[i] = bin(i).count('1')  
5     return result
```

	Test	Expected	Got	
✓	<code>print(CountingBits(2))</code>	<code>[0, 1, 1]</code>	<code>[0, 1, 1]</code>	✓
✓	<code>print(CountingBits(5))</code>	<code>[0, 1, 1, 2, 1, 2]</code>	<code>[0, 1, 1, 2, 1, 2]</code>	✓

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

```
1 def median(a, b, c):  
2     nums = [a, b, c]  
3     nums.sort()  
4     return nums[1]
```

	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 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](#)

```
1 def isPrime(n):
2     if n <= 1:
3         return False
4     elif n <= 3:
5         return True
6     elif n % 2 == 0 or n % 3 == 0:
7         return False
8     i = 5
9     while i * i <= n:
10        if n % i == 0 or n % (i + 2) == 0:
11            return False
12        i += 6
13    return True
14
15
```

	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 4

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     depth = 0
3     for char in s:
4         if char == '(':
5             depth += 1
6         elif char == ')':
7             depth -= 1
8             if depth < 0:
9                 return False
10    return depth == 0
```

	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 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

return 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 %)

```
1 def euclid_gcd(a, b):
2     if b == 0:
3         return a
4     else:
5         return euclid_gcd(b, a % b)
6
7 a = int(input())
8 b = int(input())
9 print(euclid_gcd(a, b))
```

	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...

[WEEK-07-Extra ▶](#)