

Assignment02_4086741 (CDS2003)

Q01:

Please use recursion to define and test a function to calculate the sum of a list of numbers.

A01:

The screenshot shows a code editor interface with a Python file named 'assignment02_Q01_4086741'. The code defines a recursive function 'list_sum_recur' that calculates the sum of a list. It includes test cases for a small list, a large range, and a multiplication test. The output console shows the results of these tests.

```
1 import sys
2 sys.setrecursionlimit(999999)
3
4 def list_sum_recur(num_list):
5     # Implementation here
6     if len(num_list) == 1:
7         return num_list[0]
8     else:
9         return num_list[0] + list_sum_recur(num_list[1:])
10
11 # test here
12 def test_list_sum_recur():
13     print(list_sum_recur([1, 2, 3, 4, 5]))
14     print((1+5)*5 / 2)
15
16     print(list_sum_recur(list(range(1,9999+1))))
17     print((1+9999)*9999 / 2)
18
19 test_list_sum_recur()
20
```

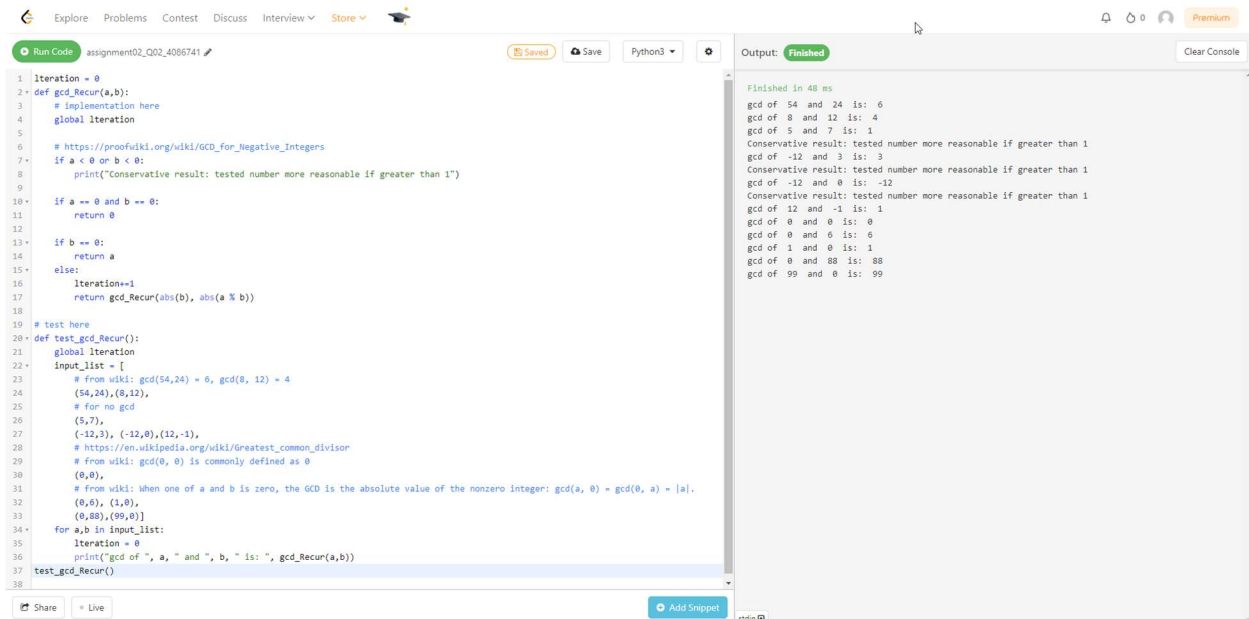
Output: **Finished**

```
Finished in 307 ms
15
15.0
49995000
49995000.0
```

Q02:

Please use recursion to define and test a function to find the greatest common division of two positive integers.

A02:



The screenshot shows a code editor with a Python3 file named 'assignment02_Q02_4086741'. The code defines a recursive function `gcd_Recur(a,b)` that calculates the greatest common divisor of two integers. It includes a test function `test_gcd_Recur()` that runs several test cases. The output console shows the results of these tests, including recursive calls and the final GCD values.

```
1 iteration = 0
2 def gcd_Recur(a,b):
3     # implementation here
4     global iteration
5
6     # https://proofwiki.org/wiki/GCD_for_Negative_Integers
7     if a < 0 or b < 0:
8         print("Conservative result: tested number more reasonable if greater than 1")
9
10    if a == 0 and b == 0:
11        return 0
12
13    if b == 0:
14        return a
15    else:
16        iteration+=1
17        return gcd_Recur(abs(b), abs(a % b))
18
19 # test here
20 def test_gcd_Recur():
21     global iteration
22     input_list = [
23         # from wiki: gcd(54,24) = 6, gcd(0, 12) = 12
24         (54,24),(0,12),
25         # for no gcd
26         (5,7),
27         (-12,3), (-12,0),(12,-1),
28         # https://en.wikipedia.org/wiki/Greatest_common_divisor
29         # from wiki: gcd(0, 0) is commonly defined as 0
30         (0,0),
31         # from wiki: when one of a and b is zero, the GCD is the absolute value of the nonzero integer: gcd(a, 0) = gcd(0, a) = |a|.
32         (0,6),(1,0),
33         (0,88),(99,0)]
34     for a,b in input_list:
35         iteration = 0
36         print("gcd of ", a, " and ", b, " is: ", gcd_Recur(a,b))
37     test_gcd_Recur()
38
```

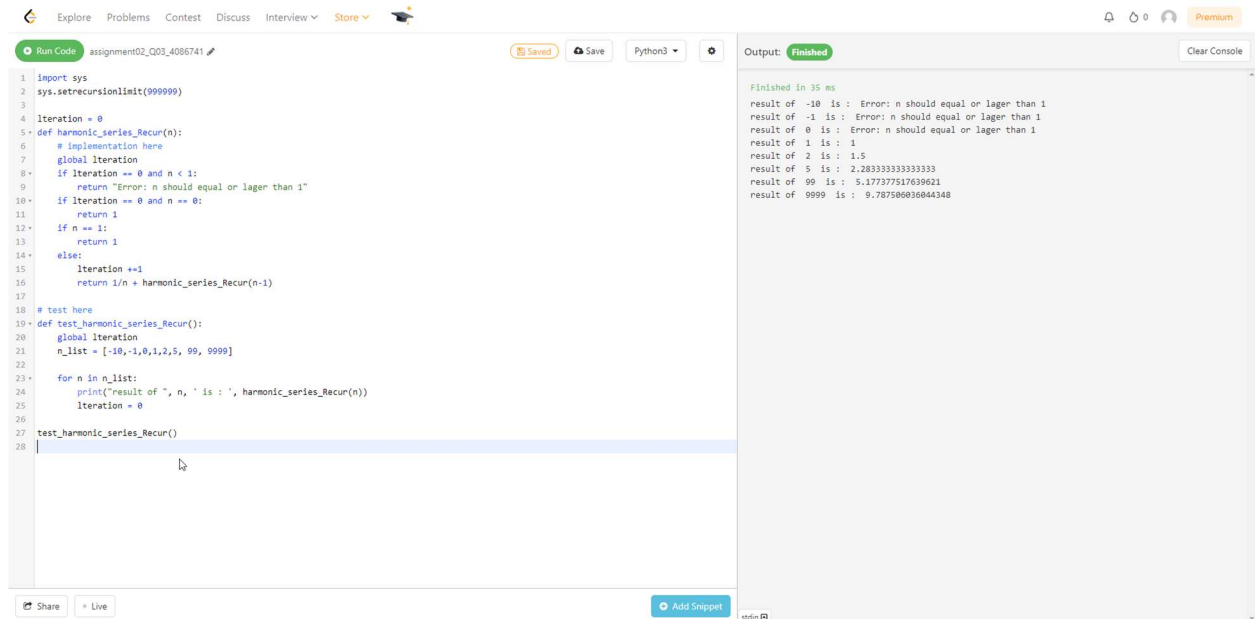
Output: Finished in 48 ms

```
gcd of 54 and 24 is: 6
gcd of 0 and 12 is: 12
gcd of 5 and 7 is: 1
Conservative result: tested number more reasonable if greater than 1
gcd of -12 and 3 is: 3
Conservative result: tested number more reasonable if greater than 1
gcd of -12 and 0 is: -12
Conservative result: tested number more reasonable if greater than 1
gcd of 12 and -1 is: 1
gcd of 0 and 0 is: 0
gcd of 0 and 6 is: 6
gcd of 1 and 0 is: 1
gcd of 0 and 88 is: 88
gcd of 99 and 0 is: 99
```

Q03:

Please use recursion to define and test a function to calculate the harmonic series up to `n` terms.

A03:



The screenshot shows a code editor with a Python3 file named 'assignment02_Q03_4086741'. The code defines a recursive function 'harmonic_series_Recur(n)' that calculates the harmonic series. It includes a global 'iteration' variable and a test function 'test_harmonic_series_Recur()' that prints the results for a list of values: [-10, -1, 0, 1, 2, 5, 99, 9999]. The output panel shows the results of these calculations, including error messages for negative values and the final sum for 9999 terms.

```
1 import sys
2 sys.setrecursionlimit(999999)
3
4 iteration = 0
5 def harmonic_series_Recur(n):
6     # Implementation here
7     global iteration
8     if iteration == 0 and n < 1:
9         return "Error: n should equal or larger than 1"
10    if iteration == 0 and n == 0:
11        return 1
12    if n == 1:
13        return 1
14    else:
15        iteration += 1
16        return 1/n + harmonic_series_Recur(n-1)
17
18 # test here
19 def test_harmonic_series_Recur():
20     global iteration
21     n_list = [-10, -1, 0, 1, 2, 5, 99, 9999]
22
23     for n in n_list:
24         print("result of ", n, " is : ", harmonic_series_Recur(n))
25         iteration = 0
26
27 test_harmonic_series_Recur()
28
```

Output: Finished in 35 ms

```
result of -10 is : Error: n should equal or larger than 1
result of -1 is : Error: n should equal or larger than 1
result of 0 is : Error: n should equal or larger than 1
result of 1 is : 1
result of 2 is : 1.5
result of 5 is : 2.283333333333333
result of 99 is : 5.177377517639621
result of 9999 is : 9.787506036044348
```

A04:

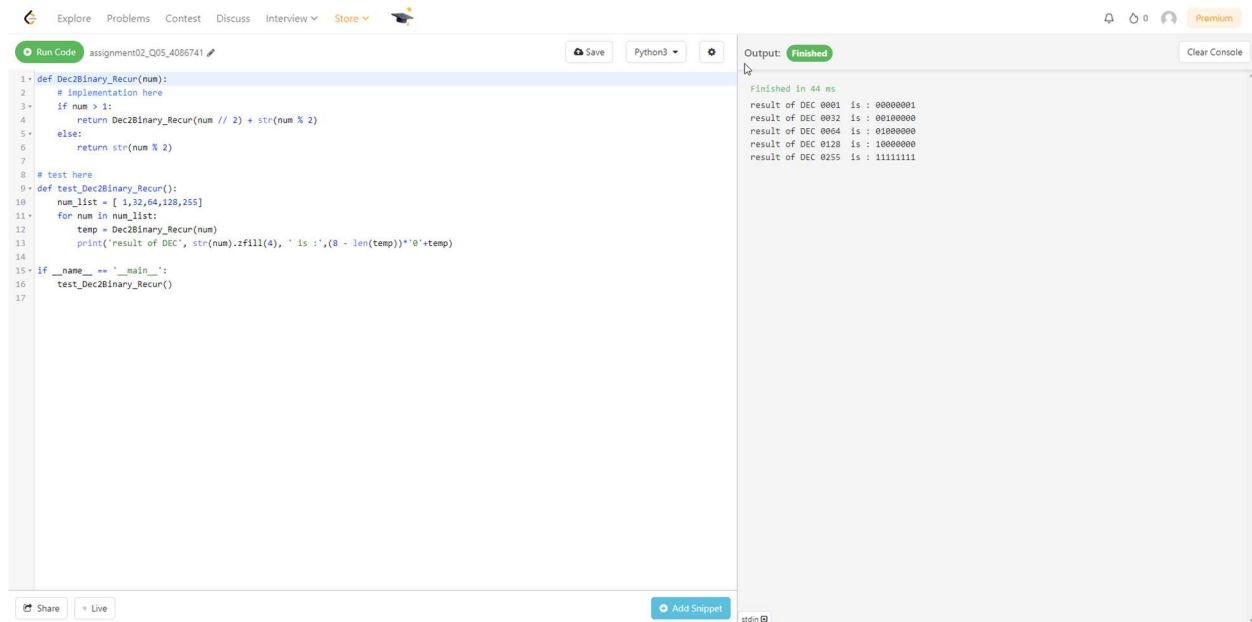
[illegible]

Optional

Q05:

Please use recursion to define and test a function to accept a decimal integer and display its binary equivalent.

A05:



The screenshot shows a code editor with a Python file named 'assignment02_Q05_4086741'. The code defines a recursive function 'Dec2Binary_Recur' that converts a decimal number to its binary string representation. The function uses the modulo 2 operation to build the binary string from least significant to most significant bits. A test function 'test_Dec2Binary_Recur' is also present, which iterates over a list of decimal numbers and prints the result of the conversion for each. The output console shows the execution results for these test cases.

```
1 def Dec2Binary_Recur(num):
2     # Implementation here
3     if num > 1:
4         return Dec2Binary_Recur(num // 2) + str(num % 2)
5     else:
6         return str(num % 2)
7
8 # test here
9 def test_Dec2Binary_Recur():
10     num_list = [1, 32, 64, 128, 255]
11     for num in num_list:
12         temp = Dec2Binary_Recur(num)
13         print("result of DEC", str(num).zfill(4), " is :", (8 - len(temp)) * '0' + temp)
14
15 if __name__ == '__main__':
16     test_Dec2Binary_Recur()
17
```

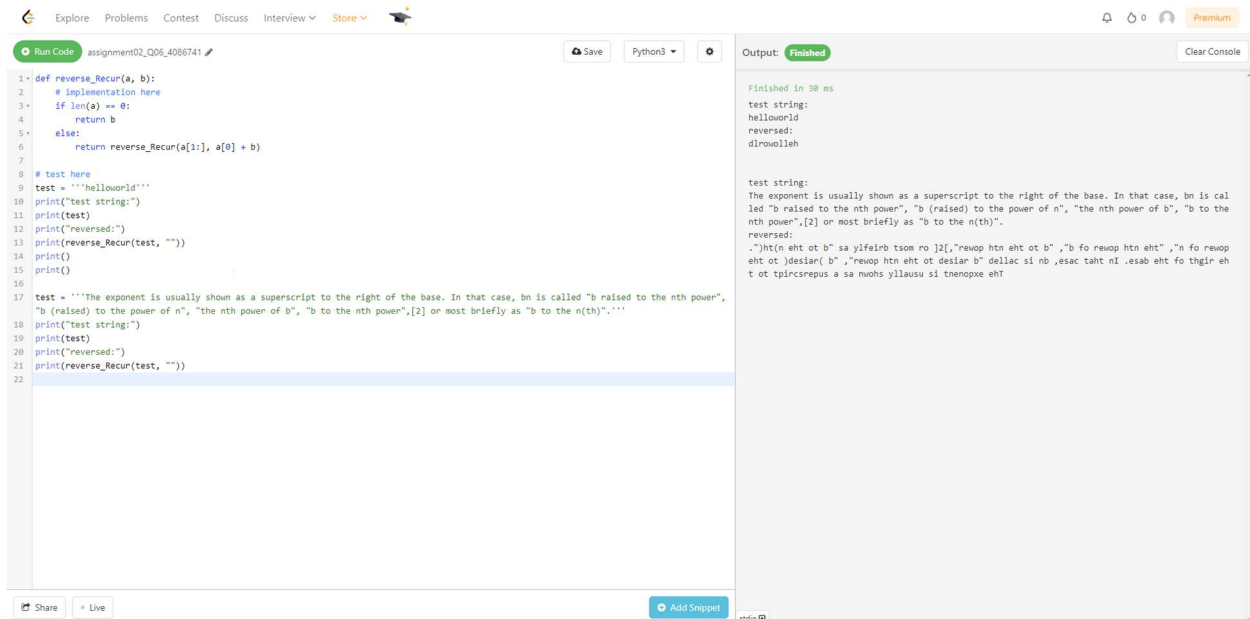
Output: Finished in 44 ms

Decimal	Binary
DEC 0001	1s : 00000001
DEC 0032	1s : 00100000
DEC 0064	1s : 01000000
DEC 0128	1s : 10000000
DEC 0255	1s : 11111111

Q06:

Please use recursion to define and test a function to take in a string and returns a reversed copy of the string.

A06:



The screenshot shows a code editor with a Python file named 'assignment02_Q06_4086741'. The code defines a recursive function 'reverse_Recur(a, b)' that reverses a string. It includes test cases for 'helloworld' and a long string. The output panel shows the results of the function calls.

```
1 def reverse_Recur(a, b):
2     # Implementation here
3     if len(a) == 0:
4         return b
5     else:
6         return reverse_Recur(a[1:], a[0] + b)
7
8 # test here
9 test = 'helloworld'
10 print("test string:")
11 print(test)
12 print("reversed:")
13 print(reverse_Recur(test, ""))
14 print()
15 print()
16
17 test = '''The exponent is usually shown as a superscript to the right of the base. In that case, bn is called "b raised to the nth power",
18 "b (raised) to the power of n", "the nth power of b", "b to the nth power",[2] or most briefly as "b to the n(th)".'''
19 print("test string:")
20 print(test)
21 print("reversed:")
22 print(reverse_Recur(test, ""))
```

Output: **Finished**

Finished in 30 ms

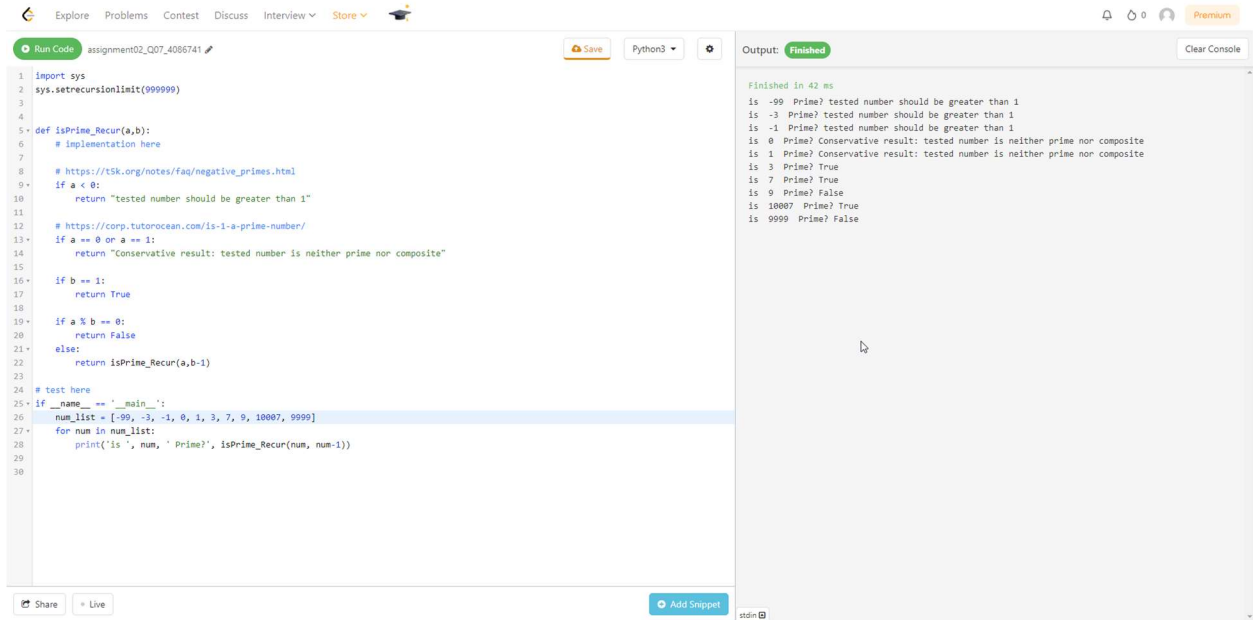
test string:
helloworld
reversed:
dlrowolleh

test string:
The exponent is usually shown as a superscript to the right of the base. In that case, bn is called "b raised to the nth power", "b (raised) to the power of n", "the nth power of b", "b to the nth power",[2] or most briefly as "b to the n(th)".
reversed:
.)ht(n eht ot b" sa ylfeirb tson ro]l["rewoptn eht ot b" ,"b fo rewoptn eht" ,"n fo rewoptn eht ot)desiar(b" ,"rewoptn eht ot desiar b" dellac si nb ,esac taht ni .esab eht fo thgir eh t ot tpircsrepus a sa muohs yllausu si tnenopxe eht

Q07:

Please use recursion to define and test a function to check whether a number is Prime or not.

A07:



The screenshot shows a code editor with a Python3 file named 'assignment02_Q07_4086741'. The code defines a recursive function 'isPrime_Recur(a,b)' to check if a number is prime. The function includes comments for negative numbers, base cases (0, 1), and a recursive step. A test list 'num_list' is provided, and the function is called for each number in the list. The output shows the results for each number, including error messages for non-positive numbers and 'Conservative result' for 0 and 1.

```
1 import sys
2 sys.setrecursionlimit(999999)
3
4
5 def isPrime_Recur(a,b):
6     # Implementation here
7
8     # https://csk.org/notes/faq/negative_primes.html
9     if a < 0:
10         return "tested number should be greater than 1"
11
12     # https://corp.tutorocean.com/is-1-a-prime-number/
13     if a == 0 or a == 1:
14         return "Conservative result: tested number is neither prime nor composite"
15
16     if b == 1:
17         return True
18
19     if a % b == 0:
20         return False
21     else:
22         return isPrime_Recur(a,b-1)
23
24 # test here
25 if __name__ == '__main__':
26     num_list = [-99, -3, -1, 0, 1, 3, 7, 9, 10007, 9999]
27     for num in num_list:
28         print('is ', num, ' Prime?', isPrime_Recur(num, num-1))
29
30
```

Output: Finished in 42 ms

```
is -99 Prime? tested number should be greater than 1
is -3 Prime? tested number should be greater than 1
is -1 Prime? tested number should be greater than 1
is 0 Prime? Conservative result: tested number is neither prime nor composite
is 1 Prime? Conservative result: tested number is neither prime nor composite
is 3 Prime? True
is 7 Prime? True
is 9 Prime? False
is 10007 Prime? True
is 9999 Prime? False
```