CSPB 2824 - Stade - Discrete Structures

Dashboard / My courses / 2237:CSPB 2824 / OPTIONAL Bonus Week / Functional Programming Project

Started on	Friday, 8 December 2023, 8:03 PM
State	Finished
Completed on	Friday, 8 December 2023, 10:34 PM
Time taken	2 hours 30 mins
Grade	**99.20** out of 100.00

Question **1**

Correct

Mark 5.00 out of 5.00

Write a function in python called

```
compute_squared_distance(x1, y1, z1, x2, y2, z2)
```

that given points (x_1, y_1, z_1) and (x_2, y_2, z_2) computes the squared distance between them using the formula

$(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2$

Notes:

- Note the name of the function and order of its arguments.
- The power operator in python is **. To square a number k, you type k ** 2
- Your code must pass all the tests. You are allowed a penalty free pre-check to debug your code.

For example:

Test	Result
`print(compute_squared_distance(0,0,0,1,1,1))`	3
`print(compute_squared_distance(2,2,2,5,6,7))`	50

Answer: (penalty regime: 10, 20, ... %)

```python
def compute_squared_distance(x1, y1, z1, x2, y2, z2):
    X = pow((x2 - x1),2)
    Y = pow((y2 - y1),2)
    Z = pow((z2 - z1),2)
    return X + Y + Z
```

	Test	Expected	Got	
✓	`print(compute_squared_distance(0,0,0,1,1,1))`	3	3	✓
✓	`print(compute_squared_distance(2,2,2,5,6,7))`	50	50	✓
✓	`print(compute_squared_distance(0,0,0,0,0,0))`	0	0	✓
✓	`print(compute_squared_distance(0,1,0,0,0,1))`	2	2	✓
✓	`print(compute_squared_distance(0,1,0,-2,0,1))`	6	6	✓
✓	`print(compute_squared_distance(0,-1,1,-2,0,1))`	5	5	✓
✓	`print(compute_squared_distance(0,0,0, 0, 0, 10))`	100	100	✓

Passed all tests! ✓

Question author's solution (Python3):

```python
def compute_squared_distance(x1, y1, z1, x2, y2, z2):
    m1 = (x1 - x2 ) **2
    m2 = (y1 - y2) ** 2
    m3 = (z1 - z2) ** 2
    return m1+m2+m3

```

Correct

Marks for this submission: 5.00/5.00.

Question **2**

Correct

Mark 8.00 out of 8.00

You are asked to design a function

```
count_all_factors(n)
```

where n is an input integer >= 1

You should return the number of factors of n, including 1 and n themselves.

Example:

Input: n = 20, Output: 6

Input n = 23, Output: 2

Input n = 35, Output: 4

For example:

Test	Result
`print(count_all_factors(20))`	6
`print(count_all_factors(23))`	2
`print(count_all_factors(35))`	4

Answer: (penalty regime: 10, 20, ... %)

```python
def count_all_factors(n):
    factors = 0
    for i in range(1, n + 1):
        if (n % i == 0):
            factors += 1
    return factors
```

	Test	Expected	Got	
✓	`print(count_all_factors(16))`	5	5	✓
✓	`print(count_all_factors(17))`	2	2	✓
✓	`print(count_all_factors(1980))`	36	36	✓
✓	`print(count_all_factors(223))`	2	2	✓
✓	`print(count_all_factors(16653))`	16	16	✓

Passed all tests! ✓

Question author's solution (Python3):

```
def count_all_factors(n):
    if n < 0:
        n = -n
    assert n >= 1
    count = 2
    for i in range(2,n):
        if n % i == 0:
            count = count + 1
    return count
```

Correct

Marks for this submission: 8.00/8.00.

Question **3**

Correct

Mark 3.00 out of 3.00

Consider the function below in python:

```
def my_little_function(a, b, c):
        x = c
        for i in range(0, a + b):
             x = x + c
        return x
```

Select all possible types that apply for this function. You are advised to write this function out into a python interpreter and test before you commit to answers. To obtain full credit you should select all the appropriate types that are possible and none of the selected types should potentially cause a runtime error.

Select one or more:

- [✓ a. int * int * int -> int]
- [✓ b. int * int * string -> string]
- [✓ c. int * int * float -> float]
- d. float * float * float -> float
- e. float * float * string -> string
- f. string * string * string -> string

Your answer is correct.

The correct answers are: int * int * int -> int, int * int * string -> string, int * int * float -> float

Correct

Marks for this submission: 3.00/3.00.

Question **4**

Correct

Mark 2.00 out of 2.00

Consider the two functions below:

```
def fun1(a, b):
       if a >= 0:
              return b + ':' + str(a)
       else:
               return b

def fun2(a):
        return a * a
```

Suppose we call

```
fun1( fun2( -25), 'world' )
```

We wish to ascertain which of the two functions is invoked first: fun1 or fun2. With this in mind, which of the following statements is true?

Select one:

[✓ a. fun2 is called first with argument -25 and then fun1 is called using its result.]

b. fun1 is called first and it then calls fun2 to obtain the value of its first argument.

c. It depends: python can call fun1 or fun2 depending on which one was declared first in the file.

d. fun2 is called first using the argument 'hello' and fun1 is called on its result

e. All of the provided answers are false.

f. All of the provided options could be true depending on the circumstances.

Your answer is correct.

The correct answer is: fun2 is called first with argument -25 and then fun1 is called using its result.

Correct

Marks for this submission: 2.00/2.00.

Question **5**

Correct

Mark 8.00 out of 8.00

You are provided two functions:

```
square(x)
```

takes a number x and computes x^2

```
multiply(x, y)
```

multiplies two numbers x and y

Write a function

```
to_the_power_seven(x)
```

that given x computes x^7

Your code should be made entirely of calls to the provided functions multiply and square. In particular, the use of loops and conditional statements will result in penalties.

For example:

Test	Result
`print(to_the_power_seven(1))`	1
`print(to_the_power_seven(2))`	128
`print(to_the_power_seven(3))`	2187

Answer: (penalty regime: 10, 20, ... %)

Reset answer

```
# please do not edit
square = lambda x: x**2
multiply = lambda x, y: x*y
# your code goes below
# Please note: your answer should be a "traditional" python function
#              using def and not a lambda.
def to_the_power_seven(n):
    return int(pow(square(multiply(n,1)),3.5))
```

	Test	Expected	Got	
✓	`print(to_the_power_seven(3))`	2187	2187	✓

	Test	Expected	Got	
✓	`print(to_the_power_seven(0))`	0	0	✓
✓	`print(to_the_power_seven(4))`	16384	16384	✓
✓	`print(to_the_power_seven(5))`	78125	78125	✓

Passed all tests! ✓

Question author's solution (Python3):

```python
square = lambda x: x**2
multiply = lambda x, y: x*y

def to_the_power_seven(x):
    x4 = square(square(x))
    x2 = square(x)
    x6 = multiply(x4, x2)
    x7 = multiply(x6, x)
    return x7

```

Correct

Marks for this submission: 8.00/8.00.

Question **6**

Correct

Mark 5.00 out of 5.00

In the answer box, you will see that we have defined three functions using lambdas:

```
add_and_multiply2(x, y, z)

is_divisible_by2(n, k)

is_empty_list2 (l)
```

Please write equivalent functions called

```
add_and_multiply(x, y, z)

is_divisible_by(n, k)

is_empty_list(l)
```

using the standard (traditional) definition using the **def** keyword.

For example:

Test	Result
`print(add_and_multiply(1, 2, 3))`	7
`print(is_divisible_by(7, 5))`	False
`print(is_divisible_by(10, 5))`	True
`print(is_empty_list([]))`	True
`print(is_empty_list(['blah','blah','blah']))`	False

Answer: (penalty regime: 10, 20, ... %)

Reset answer

```python
add_and_multiply2 = lambda x, y, z: x + y * z
is_divisible_by2 = lambda n, k: n % k == 0
is_empty_list2 = lambda l: len(l) == 0
# Do not edit above

def add_and_multiply(x, y, z):
    return x + y * z

def is_divisible_by(n, k):
    if (n % k == 0):
        return True
    else:
        return False

def is_empty_list(l):
    if (len(l) == 0):
        return True
    else:
        return False;
```

	Test	Expected	Got	
✓	`print(add_and_multiply(1, 2, 3))`	7	7	✓
✓	`print(add_and_multiply('hello', 'there', 3))`	hellotheretherethere	hellotheretherethere	✓
✓	`print(is_divisible_by(7, 5))`	False	False	✓
✓	`print(is_divisible_by(10, 5))`	True	True	✓
✓	`print(is_empty_list([]))`	True	True	✓
✓	`print(is_empty_list(['blah','blah','blah']))`	False	False	✓

Passed all tests! ✓

Question author's solution (Python3):

```python
add_and_multiply2 = lambda x, y, z: x + y * z
is_divisible_by2 = lambda n, k: n % k == 0
is_empty_list2 = lambda l: len(l) == 0
# Do not edit above

def add_and_multiply(x, y, z):
    return x + y * z

def is_divisible_by(n, k):
    return n % k == 0

def is_empty_list(l):
    return len(l) == 0

```

Correct

Marks for this submission: 5.00/5.00.

Question **7**

Correct

Mark 8.00 out of 8.00

We have provided some functions in the standard (or what we have been calling traditional) python.

```
def concat(x, y)
def concat_with_sep(x, y, sep)
def concat_friendly(x)
```

Write down equivalent functions using the lambda notation.

The names of the corresponding functions defined as lambdas should be

```
concat2(x, y)
concat_with_sep2(x, y, sep)
concat_friendly2(x)
```

You must use lambdas in your answer. Not defining them as lambdas may pass the tests but invite penalties.

For example:

Test	Result
`print(concat2('a','b'))`	`a b`
`print(concat_with_sep2('x','x','_'))`	`x_x`
`print(concat_friendly2('World'))`	`Hello: World`

Answer: (penalty regime: 10, 20, ... %)

Reset answer

```python
# DO NOT EDIT
def concat(x, y):
    return x + ' ' + y

def concat_with_sep(x, y, sep):
    return x + sep + y

def concat_friendly(x):
    return 'Hello: '+ x

# YOUR ANSWERS BELOW

concat2 = lambda x, y: x + ' ' + y
concat_with_sep2 = lambda x, y, sep: x + sep + y
concat_friendly2 = lambda x: 'Hello: ' + x
```

	Test	Expected	Got	
✓	`print(concat2('a','b'))`	`a b`	`a b`	✓
✓	`print(concat_with_sep2('x','x','_'))`	`x_x`	`x_x`	✓
✓	`print(concat_friendly2('World'))`	`Hello: World`	`Hello: World`	✓
✓	`print(concat2('Times','They Are A Changing'))`	`Times They Are A Changing`	`Times They Are A Changing`	✓

	Test	Expected	Got	
✓	`print(concat_with_sep2('a','b',','))`	`a,b`	`a,b`	✓
✓	`print(concat_friendly2('stranger!'))`	`Hello: stranger!`	`Hello: stranger!`	✓

Passed all tests! ✓

Question author's solution (Python3):

```python
def concat(x, y):
    return x + ' ' + y

def concat_with_sep(x, y, sep):
    return x + sep + y

def concat_friendly(x):
    return 'Hello: '+x # please pay attention to the space after the ":"


concat2 = lambda x, y: x +' ' + y
concat_with_sep2 = lambda x, y, sep: x + sep + y
concat_friendly2 = lambda x: 'Hello: '+ x
```

Correct

Marks for this submission: 8.00/8.00.

Question **8**

Correct

Mark 7.20 out of 8.00

Write a higher order function

```
compose_and_subtract(f, g, x)
```

where f, g are functions from natural numbers to natural numbers.
The function compose_and_subtract(x) should compute $f(g(x)) - g(f(x))$

For example:

Test	Result
`print(compose_and_subtract(square, decr, 2))`	-2
`print(compose_and_subtract(incr, decr, 14))`	0
`print(compose_and_subtract(incr, decr, 12))`	0
`print(compose_and_subtract(doub, decr, 14))`	-1

Answer: (penalty regime: 10, 20, ... %)

Reset answer

```python
# do not edit below
square = lambda x: x**2
add_two = lambda x : x + 2
decr = lambda x : x - 1
incr = lambda x : x + 1
doub = lambda x: 2 * x
# do not edit above

def compose_and_subtract(f, g, x):
    return f(g(x)) - g(f(x))
```

	Test	Expected	Got	
✓	`print(compose_and_subtract(square, decr, 2))`	-2	-2	✓
✓	`print(compose_and_subtract(square, decr, 1))`	0	0	✓
✓	`print(compose_and_subtract(square, decr, 5))`	-8	-8	✓
✓	`print(compose_and_subtract(square, decr, 14))`	-26	-26	✓
✓	`print(compose_and_subtract(incr, decr, 14))`	0	0	✓
✓	`print(compose_and_subtract(incr, decr, 12))`	0	0	✓
✓	`print(compose_and_subtract(doub, decr, 14))`	-1	-1	✓

	Test	Expected	Got	
✓	`print(compose_and_subtract(add_two, doub, 14))`	-2	-2	✓
✓	`print(compose_and_subtract(add_two, square, 12))`	-50	-50	✓

Passed all tests! ✓

Question author's solution (Python3):

```
# do not edit below
square = lambda x: x**2
add_two = lambda x : x + 2
decr = lambda x : x - 1
incr = lambda x : x + 1
doub = lambda x: 2 * x
# do not edit above

compose_and_subtract = lambda f, g, x: f(g(x)) - g(f(x))

```

Correct

Marks for this submission: 8.00/8.00. Accounting for previous tries, this gives **7.20/8.00**.

Question **9**

Correct

Mark 9.00 out of 9.00

We would like you to write a higher order function

```
apply_three_times( f,  x, y )
```

Wherein $f(a, b)$ is a function with two arguments a, b.

The function apply_three_times should compute the value of

$$f(x, f(x, f(x, y)))$$

You can write apply_three_times either as a lambda or as a function defined the standard way using the def keyword.

For example:

Test	Result
`print(apply_three_times(multiply,1,2))`	2
`print(apply_three_times(multiply,2,2))`	16
`print(apply_three_times(multiply,2,1))`	8

Answer: (penalty regime: 10, 20, ... %)

Reset answer

```
# do not edit the definitions below

multiply = lambda x, y: x * y

divide = lambda x, y: x // y

help_fun = lambda x, y: 'Help' + str(x) + str(y)

lst_ref = lambda x , y: x[y]

# do not edit the definitions above
def apply_three_times(f, x, y):
    return f(x,f(x,f(x,y)))
```

	Test	Expected	Got
✓	`print(apply_three_times(multiply,1,2))`	2	2
✓	`print(apply_three_times(multiply,2,2))`	16	16
✓	`print(apply_three_times(multiply,2,1))`	8	8
✓	`print(apply_three_times(multiply,2,'blah'))`	blahblahblahblahblahblahblahblah	blahblahblahblahblahblahbla

	Test	Expected	Got
✓	`print(apply_three_times(lst_ref, [0, 6, 1, 2, 5 , 4, 3], 2))`	3	3
✓	`print(apply_three_times(divide, 197, 3))`	65	65

Passed all tests! ✓

Question author's solution (Python3):

```python
# do not edit the definitions below

multiply = lambda x, y: x * y

divide = lambda x, y: x // y

help_fun = lambda x, y: 'Help' + str(x) + str(y)

lst_ref = lambda x , y: x[y]

# do not edit the definitions above

def apply_three_times( f,  x, y ):
    return f(x, f(x, f(x, y)))

```

Correct

Marks for this submission: 9.00/9.00.

Question **10**

Correct

Mark 5.00 out of 5.00

In this question, we ask you to write a function

```
make_friendly_list(l)
```

Given a list l of strings, you should return a new list that has all the strings in the original list with the string 'hello: ' (with one space at the end) added in front of each string in the list l.
FYI:

```
'hello:  ' + s
```

pre-pends the string 'hello: ' in front of the string s in python.
See examples provided for more examples.

You may use a loops to iterate through the string.

For example:

Test	**Result**
`print(make_friendly_list([]))`	`[]`
`print(make_friendly_list(['doggy']))`	`['hello: doggy']`
`print(make_friendly_list(['fox', 'hound']))`	`['hello: fox', 'hello: hound']`

Answer: (penalty regime: 10, 20, ... %)

```python
def make_friendly_list(l):
    if (len(l) == 0):
        return l
    else:
        ret = []
        for i in range(len(l)):
            stringVal = str(l[i])
            ret.append('hello: ' + stringVal)
        return ret
```

	Test	**Expected**	**Got**	
✓	`print(make_friendly_list([]))`	`[]`	`[]`	✓
✓	`print(make_friendly_list(['doggy']))`	`['hello: doggy']`	`['hello: doggy']`	✓
✓	`print(make_friendly_list(['fox', 'hound']))`	`['hello: fox', 'hello: hound']`	`['hello: fox', 'hello: hound']`	✓

	Test	Expected	Got	
✓	`print(make_friendly_list(['fox', 'hound', 'donkey']))`	`['hello: fox', 'hello: hound', 'hello: donkey']`	`['hello: fox', 'hello: hound', 'hello: donkey']`	✓
✓	`print(make_friendly_list(['csci', '2824', 'world']))`	`['hello: csci', 'hello: 2824', 'hello: world']`	`['hello: csci', 'hello: 2824', 'hello: world']`	✓

Passed all tests! ✓

Question author's solution (Python3):

```python
def make_friendly_list(l):
    ret_list = []
    for elts in l:
        ret_list.append('hello: '+ elts)
    return ret_list

```

Correct

Marks for this submission: 5.00/5.00.

Question **11**

Correct

Mark 5.00 out of 5.00

You are asked to write a python function

```
sum_max_and_min(l)
```

that given a list l of integers returns a number k that is the
sum of the maximum and minimum elements in l, if the list is nonempty.

If, however, l is the empty list, your function should return 0.

You may assume that all elements of the list l are between -1000 and 1000

You should not be using the inbuilts max and min in python.

For example:

Test	Result
`print(sum_max_and_min([]))`	0
`print(sum_max_and_min([2]))`	4
`print(sum_max_and_min([-2, 2]))`	0

Answer: (penalty regime: 10, 20, ... %)

```python
def sum_max_and_min(l):
    if (len(l) == 0):
        return 0
    else:
        minVal = 1000
        maxVal = -1000
        for i in range(len(l)):
            if (l[i] < minVal):
                minVal = l[i]
            if (l[i] > maxVal):
                maxVal = l[i]
        return minVal + maxVal
```

	Test	Expected	Got	
✓	`print(sum_max_and_min([]))`	0	0	✓
✓	`print(sum_max_and_min([2]))`	4	4	✓
✓	`print(sum_max_and_min([-2, 2]))`	0	0	✓
✓	`print(sum_max_and_min([-2, 0, 1, -3, 2, 5, 1]))`	2	2	✓
✓	`print(sum_max_and_min([0, 0, 0, 0, 0, 0, 0]))`	0	0	✓
✓	`print(sum_max_and_min([1,1]))`	2	2	✓
✓	`print(sum_max_and_min([-1,-1,-1]))`	-2	-2	✓

Passed all tests! ✓

Question author's solution (Python3):

```python
def sum_max_and_min(l):
    if len(l) == 0:
        return 0
    max_val = -1001
    min_val =  1001
    for i in l:
        if i >= max_val:
            max_val = i
        if i <= min_val:
            min_val = i
    return max_val + min_val
```

Correct

Marks for this submission: 5.00/5.00.

Question **12**

Correct

Mark 10.00 out of 10.00

In this question you are asked to write a function to repeat every string in a given list n times much like an echo you hear up in the mountains ;-)

```
echo_every_string(l, n)
```

Here l is a list of strings and your goal is to take each string in s and repeat it n times. Your function must return a list of strings each string in l repeated n times.

Eg,

```
echo_every_string( ['hello', 'world'],  3)
```

If l is the list ['hello', 'world']

and n is 3

then your output should be

['hellohellohello', 'worldworldworld']

Note that in python if s is a string and n is a positive integer, then s * n yields the string s repeated n times.

In particular, s*0 yields the empty string.

You should be using a map operation. Usage of any for loops will incur a penalty.

NO LOOPS

For example:

Test	Result
`print(echo_every_string(['hello','world'],3))`	`['hellohellohello', 'worldworldworld']`
`print(echo_every_string([],20))`	`[]`
`print(echo_every_string(['hello','world'],0))`	`['', '']`

Answer: (penalty regime: 10, 20, ... %)

```
def echo_every_string(l, n):
    copy = n
    if (len(l) == 0):
        empty_list = []
        return empty_list
    else:
        def multString(n):
            return n * copy
        mapped = map(multString, l)
        ret = list(mapped)
        return ret
```

	Test	Expected	Got	
✓	`print(echo_every_string(['hello','world'],3))`	`['hellohellohello', 'worldworldworld']`	`['hellohellohello', 'worldworldworld']`	✓
✓	`print(echo_every_string([],20))`	`[]`	`[]`	✓
✓	`print(echo_every_string(['hello','world'],0))`	`['', '']`	`['', '']`	✓
✓	`print(echo_every_string(['abraca','dabra'],2))`	`['abracaabraca', 'dabradabra']`	`['abracaabraca', 'dabradabra']`	✓
✓	`print(echo_every_string(['a','d','c','b','x'],1))`	`['a', 'd', 'c', 'b', 'x']`	`['a', 'd', 'c', 'b', 'x']`	✓

Passed all tests! ✓

Question author's solution (Python3):

```python
def echo_every_string(l, n):
    return list(map(lambda s: s*n, l))

```

Correct

Marks for this submission: 10.00/10.00.

Question **13**

Correct

Mark 12.00 out of 12.00

We are asked to write a function to reverse the numbers in a list:

```
reverse_numbers_in_list(l)
```

Given a list l: [12, 23, 2, 19, 45], it should return a new list [21, 32, 2, 91, 54] by reversing the numbers in a list. To help you reverse a number, we have provided functions

```
number_to_string(n) # converts a number into a string
string_to_number(s)  # converts a string to a number -- if the string has non numerical characters, an error is
raised
reverse_string(s) # reverses a string
```

Your code should use the map function and should not use a loop.

Using a for loop may pass all the tests but will be penalized.

For example:

Test	Result
`print(reverse_numbers_in_list([1,2]))`	`[1, 2]`
`print(reverse_numbers_in_list([10,20]))`	`[1, 2]`
`print(reverse_numbers_in_list([101,210]))`	`[101, 12]`

Answer: (penalty regime: 10, 20, ... %)

Reset answer

```python
# do not edit below
number_to_string = lambda n: str(n)
string_to_number = lambda s: int(s)
reverse_string = lambda s: s[::-1]
# do not edit above
def reverse_numbers_in_list(l):
    map1 = map(number_to_string,l)
    ret1 = list(map1)
    map2 = map(reverse_string, ret1)
    ret2 = list(map2)
    map3 = map(string_to_number, ret2)
    ret3 = list(map3)
    return ret3
```

	Test	Expected	Got	
✓	`print(reverse_numbers_in_list([]))`	`[]`	`[]`	✓
✓	`print(reverse_numbers_in_list([1,2]))`	`[1, 2]`	`[1, 2]`	✓
✓	`print(reverse_numbers_in_list([10,20]))`	`[1, 2]`	`[1, 2]`	✓
✓	`print(reverse_numbers_in_list([101,210]))`	`[101, 12]`	`[101, 12]`	✓

	Test	Expected	Got	
✓	`print(reverse_numbers_in_list([12,20,14,35,44,124]))`	`[21, 2, 41, 53, 44, 421]`	`[21, 2, 41, 53, 44, 421]`	✓
✓	`print(reverse_numbers_in_list(list(range(20,29))))`	`[2, 12, 22, 32, 42, 52, 62, 72, 82]`	`[2, 12, 22, 32, 42, 52, 62, 72, 82]`	✓
✓	`print(reverse_numbers_in_list(list(range(209,221))))`	`[902, 12, 112, 212, 312, 412, 512, 612, 712, 812, 912, 22]`	`[902, 12, 112, 212, 312, 412, 512, 612, 712, 812, 912, 22]`	✓

Passed all tests! ✓

Question author's solution (Python3):

```python
# do not edit below
number_to_string = lambda n: str(n)
string_to_number = lambda s: int(s)
reverse_string = lambda s: s[::-1]
# do not edit above

def reverse_number(n):
    return string_to_number(reverse_string(number_to_string(n)))

def reverse_numbers_in_list(l):
    return list(map( reverse_number, l))


```

Correct

Marks for this submission: 12.00/12.00.

Question **14**

Correct

Mark 12.00 out of 12.00

We would like you to write a python program to find all palindromes in a list l:

```
find_all_palindromes(l)
```

l is a list of strings

The function should output a list of strings that belong to l and are palindromes.

Palindrome: is a string that reads the same when reversed

eg., 'madam', 'hattah', 'pieeip', 'helleh', 'papaapap'

Your code should use the filter operation in python. Using a loop will incur penalties.

For your convenience, we provide you a function to reverse a string.

NO LOOPS

For example:

Test	Result
`print(find_all_palindromes(['hello','olleh','maddam','madam']))`	`['maddam', 'madam']`
`print(find_all_palindromes(['hello','olleh']))`	`[]`
`print(find_all_palindromes([]))`	`[]`

Answer: (penalty regime: 10, 20, ... %)

Reset answer

```python
# do not edit below
reverse_str = lambda s: s[::-1]
# do not edit above
def find_all_palindromes(l):
    def IsPalindrome(string):
        return reverse_str(string) == string
    fltr = filter(IsPalindrome, l)
    ret = list(fltr)
    return ret
```

	Test	Expected	G
✓	`print(find_all_palindromes(['hello','olleh','maddam','madam']))`	`['maddam', 'madam']`	[
✓	`print(find_all_palindromes(['hello','olleh']))`	`[]`	[
✓	`print(find_all_palindromes(['peep','littttttttttttil','pooooorrrrrrrooooooop']))`	`['peep', 'littttttttttttil']`	['
✓	`print(find_all_palindromes([]))`	`[]`	[

	Test	Expected	G
✓	`print(find_all_palindromes(['havah','nolon','elle','ava','hannah','joan','ari']))`	`['havah', 'nolon', 'elle', 'ava', 'hannah']`	`[ 'e 'h`
✓	`print(find_all_palindromes(['racecar','automobile','grandprix','rider']))`	`['racecar']`	`[`

Passed all tests! ✓

Question author's solution (Python3):

```python
# do not edit below
reverse_str = lambda s: s[::-1]
# do not edit above

is_palindrome = lambda s: s == reverse_str(s)
find_all_palindromes = lambda l: list( filter( is_palindrome, l) )

```

Correct

Marks for this submission: 12.00/12.00.

Question **15**

Complete

Not graded

REQUIRED

Describe your process for completing this project.

How long did it take?

How did you break it up?

Did you have any "ah-ha" moments?

Anything else about this project.

This will not be "graded," however i**t is required to be completed to confirm your final grade.**

I have a lot of experience in programming, from my time using Python in my Physics classes and research. So almost all of these problems were really easy for me.

I think the total time that I spent on this assignment was around 3-4 hours. I pretty much did it in one shot. I've seen some of these exact problems before when practicing Leet code problems, so most of these were really familiar to me.

Like I mentioned before, I did the entire assignment in one sitting. The parts that I had to think the hardest for were the problems that used maps and filters. These concepts are still pretty abstract to me so I struggled with them at first. After viewing the resources that you posted in Moodle it was pretty easy to figure out after that.

Interestingly enough, the problem that gave me the most fits was problem 8. I originally read the problem as:

$f(g(x) - g(f(x))$

and not

$f(g(x)) - g(f(x))$.

Idiotically, even though I wasn't passing my prechecks, I submitted one go at it and then realized that I read the problem incorrectly. I then quickly fixed it and got fully points on my second submission.

Some of my functions used inner-functions, which to the graders of this might seem a little suspicious, but this is a concept that I utilized a lot in other areas of Python. Inner functions are one of my favorite aspects of Python, as they can be really versatile for complicated algorithms. I could of written these outside the main function that I was writing but I used them anyway because I love them so much :).

Thank you for giving us this choice, as I was pretty tired of studying for exams so this was a perfect substitution. I learned a lot in this class, and upon acceptance into this program I tried to get this class transferred as I took Discrete Structures for my Math minor at CMU and I am glad that CU did not transfer it in as I learned a lot this semester and am very thankful for all of your help this semester. I hope to have another class with either Dr. Stade or Cailyn as you two make this program so much more manageable than I originally thought upon entering it. Thank you again, for everything.