[CC03] FP Programming Code

Question **1**

Correct

Marked out of 1.00

Flag question

Question text

Let **lst** be a list of integer and **n** be any value, use **high-order function approach** to write function **dist**(lst,n) that returns the list of pairs of an element of lst and n.

**For example:**

| **Test** | **Result** |
| --- | --- |
| dist([1,2,3],4) | [(1, 4),(2, 4),(3, 4)] |

Answer:(penalty regime: 0 %)

1

def dist(lst, n): return list(map(lambda a : (a, n), lst))

CheckQuestion 1

Feedback

|  | **Test** | **Expected** | **Got** |  |
| --- | --- | --- | --- | --- |
|  | dist([1,2,3],4) | [(1, 4),(2, 4),(3, 4)] | [(1, 4),(2, 4),(3, 4)] |  |
|  | dist([],4) | [] | [] |  |
|  | dist([1,2,3],'a') | [(1, 'a'),(2, 'a'),(3, 'a')] | [(1, 'a'),(2, 'a'),(3, 'a')] |  |
|  | dist([3,4,1,5],6) | [(3, 6),(4, 6),(1, 6),(5, 6)] | [(3, 6),(4, 6),(1, 6),(5, 6)] |  |
|  | dist([1],'a') | [(1, 'a')] | [(1, 'a')] |  |

Passed all tests!

### Question 2

Correct

Marked out of 1.00

Remove flag

#### Question text

Let **lst** be a list of integer and **n** be any value, use **recursive approach** to write function **dist**(lst,n) that returns the list of pairs of an element of lst and n.

**For example:**

| **Test** | **Result** |
| --- | --- |
| dist([1,2,3],4) | [(1, 4),(2, 4),(3, 4)] |

Answer:(penalty regime: 0 %)

1

2

3

def dist(lst,n):

if not lst: return []

return [(lst[0], n)] + dist(lst[1:] , n)

CheckQuestion 2

#### Feedback

|  | **Test** | **Expected** | **Got** |  |
| --- | --- | --- | --- | --- |
|  | dist([1,2,3],4) | [(1, 4),(2, 4),(3, 4)] | [(1, 4),(2, 4),(3, 4)] |  |
|  | dist([],4) | [] | [] |  |
|  | dist([1,2,3],'a') | [(1, 'a'),(2, 'a'),(3, 'a')] | [(1, 'a'),(2, 'a'),(3, 'a')] |  |
|  | dist([3,4,1,5],6) | [(3, 6),(4, 6),(1, 6),(5, 6)] | [(3, 6),(4, 6),(1, 6),(5, 6)] |  |
|  | dist([1],'a') | [(1, 'a')] | [(1, 'a')] |  |

Passed all tests!

### Question 3

Correct

Marked out of 1.00

Flag question

#### Question text

Let **lst** be a list of integer and **n** be any value, use **list comprehension approach** to write function **dist**(lst,n) that returns the list of pairs of an element of lst and n.

**For example:**

| **Test** | **Result** |
| --- | --- |
| dist([1,2,3],4) | [(1, 4),(2, 4),(3, 4)] |

Answer:(penalty regime: 0 %)

1

2

def dist(lst, n):

return [(x, n) for x in lst]

CheckQuestion 3

#### Feedback

|  | **Test** | **Expected** | **Got** |  |
| --- | --- | --- | --- | --- |
|  | dist([1,2,3],4) | [(1, 4),(2, 4),(3, 4)] | [(1, 4),(2, 4),(3, 4)] |  |
|  | dist([],4) | [] | [] |  |
|  | dist([1,2,3],'a') | [(1, 'a'),(2, 'a'),(3, 'a')] | [(1, 'a'),(2, 'a'),(3, 'a')] |  |
|  | dist([3,4,1,5],6) | [(3, 6),(4, 6),(1, 6),(5, 6)] | [(3, 6),(4, 6),(1, 6),(5, 6)] |  |
|  | dist([1],'a') | [(1, 'a')] | [(1, 'a')] |  |

Passed all tests!

### Question 4

Correct

Marked out of 1.00

Flag question

#### Question text

Let lst be a list of a list of element, use **high-order function approach** to write function **flatten**(lst) that returns the list of all elements

**For example:**

| **Test** | **Result** |
| --- | --- |
| flatten([[1,2,3],[4,5],[6,7]]) | [1,2,3,4,5,6,7] |

Answer:(penalty regime: 0 %)

1

2

3

from functools import reduce

def flatten(lst):

return reduce(lambda a, b:a + b, lst, [])

CheckQuestion 4

#### Feedback

|  | **Test** | **Expected** | **Got** |  |
| --- | --- | --- | --- | --- |
|  | flatten([[1,2,3],[4,5],[6,7]]) | [1,2,3,4,5,6,7] | [1,2,3,4,5,6,7] |  |
|  | flatten([[]]) | [] | [] |  |
|  | flatten([]) | [] | [] |  |
|  | flatten([[1,2,3]]) | [1,2,3] | [1,2,3] |  |
|  | flatten([[1],[2],[3],[4],[5,6,7]]) | [1,2,3,4,5,6,7] | [1,2,3,4,5,6,7] |  |

### Question 5

Correct

Marked out of 1.00

Flag question

#### Question text

Let lst be a list of a list of element, use **list comprehension approach** to write function **flatten**(lst) that returns the list of all elements

**For example:**

| **Test** | **Result** |
| --- | --- |
| flatten([[1,2,3],[4,5],[6,7]]) | [1,2,3,4,5,6,7] |

Answer:(penalty regime: 0 %)

1

def flatten(lst):return(b for a in lst for b in a )

CheckQuestion 5

#### Feedback

|  | **Test** | **Expected** | **Got** |  |
| --- | --- | --- | --- | --- |
|  | flatten([[1,2,3],[4,5],[6,7]]) | [1,2,3,4,5,6,7] | [1,2,3,4,5,6,7] |  |
|  | flatten([[]]) | [] | [] |  |
|  | flatten([]) | [] | [] |  |
|  | flatten([[1,2,3]]) | [1,2,3] | [1,2,3] |  |
|  | flatten([[1],[2],[3],[4],[5,6,7]]) | [1,2,3,4,5,6,7] | [1,2,3,4,5,6,7] |  |

Passed all tests!

### Question 6

Correct

Marked out of 1.00

Flag question

#### Question text

Let lst be a list of a list of element, use **recursive approach** to write function **flatten**(lst) that returns the list of all elements

**For example:**

| **Test** | **Result** |
| --- | --- |
| flatten([[1,2,3],[4,5],[6,7]]) | [1,2,3,4,5,6,7] |

Answer:(penalty regime: 0 %)

1

2

3

4

5

6

7

def flatten(lst):

if not lst: return []

if isinstance(lst[0] , list):

return flatten(lst[0]) + flatten(lst[1:])

else:

return [lst[0]] + flatten(lst[1:] )

CheckQuestion 6

#### Feedback

|  | **Test** | **Expected** | **Got** |  |
| --- | --- | --- | --- | --- |
|  | flatten([[1,2,3],[4,5],[6,7]]) | [1,2,3,4,5,6,7] | [1,2,3,4,5,6,7] |  |
|  | flatten([[]]) | [] | [] |  |
|  | flatten([]) | [] | [] |  |
|  | flatten([[1,2,3]]) | [1,2,3] | [1,2,3] |  |
|  | flatten([[1],[2],[3],[4],[5,6,7]]) | [1,2,3,4,5,6,7] | [1,2,3,4,5,6,7] |  |

Passed all tests!

Top of Form

### Question 7

Correct

Marked out of 1.00

Flag question

#### Question text

Let **lst** be a list of integer and **n** be an integer, use **high-order function approach** to write function **lessThan**(lst,n) that returns the list of all numbers in **lst** less than **n**.

**For example:**

| **Test** | **Result** |
| --- | --- |
| lessThan([1,2,3,4,5],4) | [1,2,3] |

Answer:(penalty regime: 0 %)

1

2

def lessThan(lst, n ):

return list(filter(lambda a : a < n, lst))

CheckQuestion 7

#### Feedback

|  | **Test** | **Expected** | **Got** |  |
| --- | --- | --- | --- | --- |
|  | lessThan([1,2,3,4,5],4) | [1,2,3] | [1,2,3] |  |
|  | lessThan([],3) | [] | [] |  |
|  | lessThan([5,2,6,4,1],3) | [2,1] | [2,1] |  |
|  | lessThan([7,6,4,4,5],2) | [] | [] |  |
|  | lessThan([1,2,3,-1,0],6) | [1,2,3,-1,0] | [1,2,3,-1,0] |  |

Passed all tests!

Bottom of Form

### Question 8

Correct

Marked out of 1.00

Flag question

#### Question text

Let **lst** be a list of integer and **n** be an integer, use **recursive approach** to write function **lessThan**(lst,n) that returns the list of all numbers in **lst** less than **n**.

**For example:**

| **Test** | **Result** |
| --- | --- |
| lessThan([1,2,3,4,5],4) | [1,2,3] |

Answer:(penalty regime: 0 %)

1

2

3

4

def lessThan(lst, n):

if not lst: return []

return [lst[0]] + lessThan(lst[1:], n) if lst[0] < n else lessThan(lst[1:], n)

CheckQuestion 8

#### Feedback

|  | **Test** | **Expected** | **Got** |  |
| --- | --- | --- | --- | --- |
|  | lessThan([1,2,3,4,5],4) | [1,2,3] | [1,2,3] |  |
|  | lessThan([],2) | [] | [] |  |
|  | lessThan([5,2,6,4,1],3) | [2,1] | [2,1] |  |
|  | lessThan([7,6,3,3,5],3) | [] | [] |  |
|  | lessThan([1,2,3,-1,0],6) | [1,2,3,-1,0] | [1,2,3,-1,0] |  |

Passed all tests!

Top of Form

### Question 9

Correct

Marked out of 1.00

Flag question

#### Question text

Let **lst** be a list of integer and **n** be an integer, use **list comprehension approach** to write function **lessThan**(lst,n) that returns the list of all numbers in **lst** less than **n**.

**For example:**

| **Test** | **Result** |
| --- | --- |
| lessThan([1,2,3,4,5],4) | [1,2,3] |

Answer:(penalty regime: 0 %)

1

def lessThan(lst, n): return [i for i in lst if i < n]

CheckQuestion 9

#### Feedback

|  | **Test** | **Expected** | **Got** |  |
| --- | --- | --- | --- | --- |
|  | lessThan([1,2,3,4,5],4) | [1,2,3] | [1,2,3] |  |
|  | lessThan([],4) | [] | [] |  |
|  | lessThan([5,2,6,4,1],5) | [2,4,1] | [2,4,1] |  |
|  | lessThan([7,6,4,3,5],3) | [] | [] |  |
|  | lessThan([1,2,3,-1,0],4) | [1,2,3,-1,0] | [1,2,3,-1,0] |  |

Passed all tests!

Bottom of Form

### Question 10

Correct

Marked out of 1.00

Flag question

#### Question text

Use recursive approach to write a function lstSquare(n:Int) that returns a list of the squares of the numbers from 1 to n?

**For example:**

| **Test** | **Result** |
| --- | --- |
| lstSquare(3) | [1,4,9] |

Answer:(penalty regime: 0 %)

1

2

3

4

def lstSquare(n):

if n == 0: return []

else: return lstSquare(n-1) + [n\*n]

CheckQuestion 10

#### Feedback

|  | **Test** | **Expected** | **Got** |  |
| --- | --- | --- | --- | --- |
|  | lstSquare(3) | [1,4,9] | [1,4,9] |  |
|  | lstSquare(1) | [1] | [1] |  |
|  | lstSquare(5) | [1,4,9,16,25] | [1,4,9,16,25] |  |
|  | lstSquare(4) | [1,4,9,16] | [1,4,9,16] |  |

Passed all tests!

### Question 11

Correct

Marked out of 1.00

Flag question

#### Question text

Use list comprehension approach to write a function lstSquare(n:Int) that returns a list of the squares of the numbers from 1 to n?

**For example:**

| **Test** | **Result** |
| --- | --- |
| lstSquare(3) | [1,4,9] |

Answer:(penalty regime: 0 %)

1

2

def lstSquare(n):

return [i\*i for i in range(1, n + 1)]

CheckQuestion 11

#### Feedback

|  | **Test** | **Expected** | **Got** |  |
| --- | --- | --- | --- | --- |
|  | lstSquare(3) | [1,4,9] | [1,4,9] |  |
|  | lstSquare(1) | [1] | [1] |  |
|  | lstSquare(5) | [1,4,9,16,25] | [1,4,9,16,25] |  |
|  | lstSquare(4) | [1,4,9,16] | [1,4,9,16] |  |

Passed all tests!

### Question 12

Correct

Marked out of 1.00

Flag question

#### Question text

Use high-order function approach to write function lstSquare(n:Int) to return a list of i square for i from 1 to n?

**For example:**

| **Test** | **Result** |
| --- | --- |
| lstSquare(3) | [1,4,9] |

Answer:(penalty regime: 0 %)

1

def lstSquare(n ): return map(lambda a: a\*a, range(1, n+ 1))

CheckQuestion 12

#### Feedback

|  | **Test** | **Expected** | **Got** |  |
| --- | --- | --- | --- | --- |
|  | lstSquare(3) | [1,4,9] | [1,4,9] |  |
|  | lstSquare(1) | [1] | [1] |  |
|  | lstSquare(5) | [1,4,9,16,25] | [1,4,9,16,25] |  |
|  | lstSquare(4) | [1,4,9,16] | [1,4,9,16] |  |

Passed all tests!

Top of Form

### Question 13

Correct

Marked out of 1.00

Flag question

#### Question text

Scala has function compose to compose two functions but Python does not have this function. Write function **compose** that can takes at least two functions  as its parameters and returns the composition of these parameter functions. For example **compose(f,g,h)(x)**is defined as **f(g(h(x)))**.

**For example:**

| **Test** | **Result** |
| --- | --- |
| f = compose(increase,square)  print(f(3)) #increase(square(3)) = 10 | 10 |

Answer:(penalty regime: 0 %)

1

2

3

4

5

6

7

8

9

10

11

12

13

def increase(a): return a+1

def square(b): return b\*b

def double(c): return c\*2

def compose(\*func):

if len(func) < 2:

raise TypeError

def inner(arg):

for f in reversed(func):

arg = f(arg)

return arg

return inner

f = compose(increase, square, double)

CheckQuestion 13

#### Feedback

|  | **Test** | **Expected** | **Got** |  |
| --- | --- | --- | --- | --- |
|  | f = compose(increase,square)  print(f(3)) #increase(square(3)) = 10 | 10 | 10 |  |
|  | f = compose(increase,square,double)  print(f(3)) | 37 | 37 |  |
|  | f = compose(increase,square,double,decrease)  print(f(3)) | 17 | 17 |  |
|  | try:  f = compose(increase)  except TypeError:  print("compose() missing 1 required positional argument") | compose() missing 1 required positional argument | compose() missing 1 required positional argument |  |
|  | try:  f = compose()  except TypeError:  print("compose() missing 1 required positional argument") | compose() missing 1 required positional argument | compose() missing 1 required positional argument |  |

Passed all tests!

Bottom of Form