

## Homework 2

● Graded

Student

Sangwon Ji

Total Points

2 / 2 pts

Autograder Score

2.0 / 2.0

### Autograder Results

```
=====
Assignment: Homework 2
OK, version v1.18.1
=====
```

False

~~~~~
Scoring tests

-----
Doctests for product

```
>>> from hw02 import *
>>> product(3, identity) # 1 * 2 * 3
6
>>> product(5, identity) # 1 * 2 * 3 * 4 * 5
120
>>> product(3, square) # 1^2 * 2^2 * 3^2
36
>>> product(5, square) # 1^2 * 2^2 * 3^2 * 4^2 * 5^2
14400
>>> product(3, increment) # (1+1) * (2+1) * (3+1)
24
>>> product(3, triple) # 1*3 * 2*3 * 3*3
162
Score: 1.0/1
```

-----
Doctests for accumulate

```
>>> from hw02 import *
>>> accumulate(add, 0, 5, identity) # 0 + 1 + 2 + 3 + 4 + 5
15
>>> accumulate(add, 11, 5, identity) # 11 + 1 + 2 + 3 + 4 + 5
26
>>> accumulate(add, 11, 0, identity) # 11
11
>>> accumulate(add, 11, 3, square) # 11 + 1^2 + 2^2 + 3^2
```

```

25
>>> accumulate(mul, 2, 3, square) # 2 * 1^2 * 2^2 * 3^2
72
>>> # 2 + (1^2 + 1) + (2^2 + 1) + (3^2 + 1)
>>> accumulate(lambda x, y: x + y + 1, 2, 3, square)
19
>>> # ((2 * 1^2 * 2) * 2^2 * 2) * 3^2 * 2
>>> accumulate(lambda x, y: 2 * x * y, 2, 3, square)
576
>>> accumulate(lambda x, y: (x + y) % 17, 19, 20, square)
16
Score: 1.0/1

```

---

#### Doctests for funception

```

>>> from hw02 import *
>>> def func1(num):
...     return num + 1
>>> func2_1 = funception(func1, 0)
>>> func2_1(3) # func1(0) * func1(1) * func1(2) = 1 * 2 * 3 = 6
6
>>> func2_2 = funception(func1, 1)
>>> func2_2(4) # func1(1) * func1(2) * func1(3) = 2 * 3 * 4 = 24
24
>>> func2_3 = funception(func1, 3)
>>> func2_3(2) # Returns func1(3) since start >= stop
4
>>> func2_4 = funception(func1, 3)
>>> func2_4(3) # Returns func1(3) since start >= stop
4
>>> func2_5 = funception(func1, -2)
>>> func2_5(-3) # Returns None since start < 0
>>> func2_6 = funception(func1, -1)
>>> func2_6(4) # Returns None since start < 0
Score: 1.0/1

```

---

#### Doctests for num\_eights

```

>>> from hw02 import *
>>> num_eights(3)
0
>>> num_eights(8)
1
>>> num_eights(88888888)
8
>>> num_eights(2638)
1
>>> num_eights(86380)
2
>>> num_eights(12345)
0

```

```
>>> num_eights(8782089)
3
>>> from construct_check import check
>>> # ban all assignment statements
>>> check(HW_SOURCE_FILE, 'num_eights',
... ['Assign', 'AnnAssign', 'AugAssign', 'NamedExpr', 'For', 'While'])
True
Score: 1.0/1
```

---

#### Doctests for waves

```
>>> from hw02 import *
>>> waves(1)
True
>>> waves(10001)
False
>>> waves(12233121)
False
>>> waves(1313)
True
>>> waves(12332023213)
True
>>> from construct_check import check
>>> # ban all loops
>>> check(HW_SOURCE_FILE, 'waves',
... ['For', 'While'])
True
Score: 1.0/1
```

---

#### Doctests for count\_coins

```
>>> from hw02 import *
>>> count_coins(15)
6
>>> count_coins(10)
4
>>> count_coins(20)
9
>>> count_coins(100) # How many ways to make change for a dollar?
242
>>> count_coins(200)
1463
>>> from construct_check import check
>>> # ban iteration
>>> check(HW_SOURCE_FILE, 'count_coins', ['While', 'For'])
True
Score: 1.0/1
```

---

Point breakdown  
product: 1.0/1

accumulate: 1.0/1  
funception: 1.0/1  
num\_eights: 1.0/1  
waves: 1.0/1  
count\_coins: 1.0/1

Score:  
Total: 6.0

Cannot backup when running ok with --local.

-----  
Final Score:2.0

## Submitted Files

```
1  from operator import add, mul
2
3  square = lambda x: x * x
4
5  identity = lambda x: x
6
7  triple = lambda x: 3 * x
8
9  increment = lambda x: x + 1
10
11
12 HW_SOURCE_FILE = __file__
13
14
15 def product(n, term):
16     """Return the product of the first n terms in a sequence.
17
18     n: a positive integer
19     term: a function that takes one argument to produce the term
20
21     >>> product(3, identity) # 1 * 2 * 3
22     6
23     >>> product(5, identity) # 1 * 2 * 3 * 4 * 5
24     120
25     >>> product(3, square) # 1^2 * 2^2 * 3^2
26     36
27     >>> product(5, square) # 1^2 * 2^2 * 3^2 * 4^2 * 5^2
28     14400
29     >>> product(3, increment) # (1+1) * (2+1) * (3+1)
30     24
31     >>> product(3, triple) # 1*3 * 2*3 * 3*3
32     162
33     """
34     x, total = 1, 1
35     while x <= n: # Until x is bigger than n
36         x, total = x + 1, term(x) * total
37     return total
38
39
40
41 def accumulate(merger, start, n, term):
42     """Return the result of merging the first n terms in a sequence and start.
43     The terms to be merged are term(1), term(2), ..., term(n). merger is a
44     two-argument commutative function.
45
46     >>> accumulate(add, 0, 5, identity) # 0 + 1 + 2 + 3 + 4 + 5
47     15
48     >>> accumulate(add, 11, 5, identity) # 11 + 1 + 2 + 3 + 4 + 5
49     26
```

```

50 >>> accumulate(add, 11, 0, identity) # 11
51 11
52 >>> accumulate(add, 11, 3, square) # 11 + 1^2 + 2^2 + 3^2
53 25
54 >>> accumulate(mul, 2, 3, square) # 2 * 1^2 * 2^2 * 3^2
55 72
56 >>> # 2 + (1^2 + 1) + (2^2 + 1) + (3^2 + 1)
57 >>> accumulate(lambda x, y: x + y + 1, 2, 3, square)
58 19
59 >>> # ((2 * 1^2 * 2) * 2^2 * 2) * 3^2 * 2
60 >>> accumulate(lambda x, y: 2 * x * y, 2, 3, square)
61 576
62 >>> accumulate(lambda x, y: (x + y) % 17, 19, 20, square)
63 16
64 """
65 x, total = 1, start
66 while x<=n :
67     x, total = x+1, merger(term(x), total)
68 return total
69
70
71 def summation_using_accumulate(n, term):
72     """Returns the sum: term(1) + ... + term(n), using accumulate.
73
74     >>> summation_using_accumulate(5, square)
75     55
76     >>> summation_using_accumulate(5, triple)
77     45
78     >>> # You aren't expected to understand the code of this test.
79     >>> # Check that the bodies of the functions are just return statements.
80     >>> # If this errors, make sure you have removed the "****YOUR CODE HERE****".
81     >>> import inspect, ast
82     >>> [type(x).__name__ for x in
ast.parse(inspect.getsource(summation_using_accumulate)).body[0].body]
83     ['Expr', 'Return']
84     """
85     return accumulate(add, 0, n, term)
86
87
88 def product_using_accumulate(n, term):
89     """Returns the product: term(1) * ... * term(n), using accumulate.
90
91     >>> product_using_accumulate(4, square)
92     576
93     >>> product_using_accumulate(6, triple)
94     524880
95     >>> # You aren't expected to understand the code of this test.
96     >>> # Check that the bodies of the functions are just return statements.
97     >>> # If this errors, make sure you have removed the "****YOUR CODE HERE****".
98     >>> import inspect, ast
99     >>> [type(x).__name__ for x in
ast.parse(inspect.getsource(product_using_accumulate)).body[0].body]

```

```

100     ['Expr', 'Return']
101     """
102     return accumulate(mul, 1, n, term)
103
104 def funception(func1, start):
105     """ Takes in a function (function 1) and a start value.
106     Returns a function (function 2) that will find the product of
107     function 1 applied to the range of numbers from
108     start (inclusive) to stop (exclusive)
109
110     >>> def func1(num):
111     ...     return num + 1
112     >>> func2_1 = funception(func1, 0)
113     >>> func2_1(3) # func1(0) * func1(1) * func1(2) = 1 * 2 * 3 = 6
114     6
115     >>> func2_2 = funception(func1, 1)
116     >>> func2_2(4) # func1(1) * func1(2) * func1(3) = 2 * 3 * 4 = 24
117     24
118     >>> func2_3 = funception(func1, 3)
119     >>> func2_3(2) # Returns func1(3) since start >= stop
120     4
121     >>> func2_4 = funception(func1, 3)
122     >>> func2_4(3) # Returns func1(3) since start >= stop
123     4
124     >>> func2_5 = funception(func1, -2)
125     >>> func2_5(-3) # Returns None since start < 0
126     >>> func2_6 = funception(func1, -1)
127     >>> func2_6(4) # Returns None since start < 0
128     """
129     def func2(stop):
130         if start < 0:
131             return None
132         elif start >= stop:
133             return func1(start)
134         else:
135             total = 1
136             for i in range(start, stop):
137                 total = total * func1(i)
138             return total
139     return func2
140
141
142
143 def num_eights(n):
144     """Returns the number of times 8 appears as a digit of n.
145
146     >>> num_eights(3)
147     0
148     >>> num_eights(8)
149     1
150     >>> num_eights(88888888)
151     8

```

```

152 >>> num_eights(2638)
153 1
154 >>> num_eights(86380)
155 2
156 >>> num_eights(12345)
157 0
158 >>> num_eights(8782089)
159 3
160 >>> from construct_check import check
161 >>> # ban all assignment statements
162 >>> check(HW_SOURCE_FILE, 'num_eights',
163 ...      ['Assign', 'AnnAssign', 'AugAssign', 'NamedExpr', 'For', 'While'])
164 True
165 """
166 if n < 8:
167     return 0
168 else:
169     if n % 10 == 8:
170         return 1 + num_eights(n//10)
171     else:
172         return num_eights(n//10)
173
174
175 def waves(n):
176     """Return whether n is balanced.
177
178     >>> waves(1)
179     True
180     >>> waves(10001)
181     False
182     >>> waves(12233121)
183     False
184     >>> waves(1313)
185     True
186     >>> waves(12332023213)
187     True
188     >>> from construct_check import check
189     >>> # ban all loops
190     >>> check(HW_SOURCE_FILE, 'waves',
191 ...         ['For', 'While'])
192     True
193     """
194     #def crest(prev, curr, next):
195     #    #return prev < curr and next < curr
196     #def trough (prev, curr, next):
197     #    #return curr < prev and curr < next
198     #def plateau(prev, curr, next):
199     #    #return prev == curr == next
200     def helper(n, count, prev):
201         curr, next, rest = n % 10, (n // 10) % 10, n // 10
202         if n == 0:
203             return True

```



```

204     if prev > curr:
205         if curr == next:
206             # decrease stay 2
207             return False
208         else:
209             # Decrease stay Increase or Decrease Decrease Decrease
210             return True and helper(n//10 ,count +1 , n%10)
211     elif prev < curr:
212         # Increase stay stay or Increase Increase Increase, or Increease stay decrease
213         return True and helper(n//10, count +1, n%10)
214     else:
215         if curr < next:
216             # stay 2 increase
217             return False
218         else:
219             # stay stay decrease or stay x3
220             return True and helper(n // 10, count, n % 10)
221 return helper(n // 10, 0, n % 10)
222
223
224 def next_larger_coin(coin):
225     """Returns the next larger coin in order.
226     >>> next_larger_coin(1)
227     5
228     >>> next_larger_coin(5)
229     10
230     >>> next_larger_coin(10)
231     25
232     >>> next_larger_coin(2) # Other values return None
233     """
234     if coin == 1:
235         return 5
236     elif coin == 5:
237         return 10
238     elif coin == 10:
239         return 25
240
241
242 def next_smaller_coin(coin):
243     """Returns the next smaller coin in order.
244     >>> next_smaller_coin(25)
245     10
246     >>> next_smaller_coin(10)
247     5
248     >>> next_smaller_coin(5)
249     1
250     >>> next_smaller_coin(2) # Other values return None
251     """
252     if coin == 25:
253         return 10
254     elif coin == 10:
255         return 5

```

```
256     elif coin == 5:
257         return 1
258
259
260 def count_coins(total):
261     """Return the number of ways to make change using coins of value of 1, 5, 10, 25.
262     >>> count_coins(15)
263     6
264     >>> count_coins(10)
265     4
266     >>> count_coins(20)
267     9
268     >>> count_coins(100) # How many ways to make change for a dollar?
269     242
270     >>> count_coins(200)
271     1463
272     >>> from construct_check import check
273     >>> # ban iteration
274     >>> check(HW_SOURCE_FILE, 'count_coins', ['While', 'For'])
275     True
276     """
277     def combinations(amount, coin):
278         if amount == 0:
279             return 1
280         elif coin == None:
281             return 0
282         elif amount < 0:
283             return 0
284         else:
285             current = combinations( amount - coin, coin)
286             smaller = combinations (amount, next_smaller_coin(coin))
287             return current + smaller
288     return combinations(total, 25)
289
290 print(waves(1223121))
```