CS 61A Spring 2022 — Structure and Interpretation of Computer Programs

Midterm 2 Debugging Practice

INSTRUCTIONS

The questions in this document *do not* reflect the scope or contents of the actual midterm exam. This set of problems is designed solely to help you get a sense of what the debugging section of the exam might look like.

1. Catch that Bug!

Your classmates are trying to complete a coding assignment but are struggling. The assignment is to write a function `digit_counter`, which takes in `f`, a one-argument function, and `n`, a non-negative integer, and returns the number of digits in `n` for which `f(digit)` returns `True`.

Here's a doctest showing the intended behavior:

```
>>> is_even = lambda x: (x % 2) == 0
>>> digit_counter(is_even, 1112)
1
>>> digit_counter(is_even, 9843)
2
>>> greater_than_three = lambda x: x > 3
>>> digit_counter(greater_than_three, 123456789)
6
```

In all below parts of this problem, you don't need to indent your answer when writing code. Each solution can be made correct by changing exactly one line.

(a) Albert has decided to use a while loop to complete the assignment, but it's not working!

```
1 def digit_counter(f, n):
2     counter = 0
3     while n >= 0:
4         if f(n % 10):
5             counter += 1
6         n = n // 10
7     return counter
```

Which line number is Albert's error on?

3

(b) What should that line be replaced with?

n > 0

(c) Alex has decided to use recursion instead, but it's also not working!

```
1 def digit_counter(f, n):
2     if n < 10 and f(n):
3         return n
4     if f(n % 10):
5         return 1 + digit_counter(f, n // 10)
6     return digit_counter(f, n // 10)
```

Which line number is Alex's error on?

2

(d) What should that line be replaced with?

if n == 0:

(e) Catherine has taken an unconventional approach and has decided to use a helper function – and it's still wrong!

```
1 def digit_counter(f, n):
2     def helper(x, sofar):
3         if x > n:
4             return sofar
5         last = (n // x) % 10
6         return helper(x * 10, sofar + f(last))
7     return helper(0, 0)
```

Recall that if you try to add a boolean to a number, `True` gets treated as `1` and `False` gets treated as `0`.

Which line number is Catherine's error on?

7

(f) What should that line be replaced with?

return helper(1, 0)

This was Question 2 on the Summer 2021 Diagnostic Quiz.

2. Trace it back!

Some of your TAs are trying to write their own implementation of the `map` function, which takes in a one-argument function and applies it to every element in a list, returning a new list. They're running into some bugs, and need your help to figure out what's going wrong!

This is what we expect to see when we run each of the `my_map` functions below:

```
>>> lst = [1, 2, 3, 4, 5]
>>> double = lambda x: x * 2
>>> my_map(double, lst)
[2, 4, 6, 8, 10]
```

(a) This is Grace's code:

```
1 def my_map(func, seq):
2     res = []
3     for i in seq:
4         curr = func(seq[i])
5         res.append(curr)
6     return res
```

This is the error that she gets when she tries to run it:

```
>>> lst = [1, 2, 3, 4, 5]
>>> double = lambda x: x * 2
>>> my_map(double, lst)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/Users/grace/cs61a/my_map.py", line 4, in my_map
    curr = func(seq[i])
IndexError: list index out of range
```

i. According to the traceback, what line is Grace's error on?

4

ii. Rewrite the line you mentioned above in order to correct Grace's error.

```
curr = func(i)
```

Note: It is possible to diagnose an error on line 3 here. However, the question specifically asks you to point out the line that the traceback is complaining about, which is line 4. This question is designed to show you that errors can cascade – meaning, you can have an error on one line that's caused by an error on another line. You won't always be able to solve a traceback's indicated error by changing the indicated line. This is possible here, but we'll see otherwise in some subsequent questions...

(b) This is Vanshaj's code:

```
1 def my_map(func, seq):
2     if len(seq) == 0:
3         return seq
4     return func(seq[:1]) + my_map(func, seq[1:])
```

This is the result he gets when he tries to run it:

```
>>> lst = [1, 2, 3, 4, 5]
>>> double = lambda x: x * 2
>>> my_map(double, lst)
[1, 1, 2, 2, 3, 3, 4, 4, 5, 5]
```

i. What line of code is causing Vanshaj's code to produce unexpected output?

> **4**

ii. Rewrite the line you identified above to fix the error and produce the expected result.

> `return [func(seq[0])] + my_map(func, seq[1:])`

Note: Often times, your code will error by way of producing unexpected output. This could be the result of a typo somewhere, or of a logical flaw in the code. Either way, these are usually the hardest types of bugs to fix, because the program itself runs just fine. For questions like these, it helps to try to look for places where a logical error could have occurred, as well as how the resulting behavior could've occurred. Here, for instance, we notice that our output has every element occurring twice (instead of being doubled), which means that we're somehow passing in a list into our double function.

(c) This is Marie's code:

```
1 def my_map(func, seq):
2     res = {}
3     for el in seq:
4         res.append(func(el))
5     return res
```

This is the result she gets when she tries to run it:

```
>>> lst = [1, 2, 3, 4, 5]
>>> double = lambda x: x * 2
>>> my_map(double, lst)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/Users/marie/cs61a/my_map.py", line 4, in my_map
    res.append(func(el))
AttributeError: 'dict' object has no attribute 'append'
```

i. What line of code should you fix to prevent this error from happening?

2

ii. Rewrite the line you identified above to fix the error.

```
res = []
```

Note: This is the complement to Part (a), where we relied on the traceback to determine what line of code was wrong. In this part, however, the traceback doesn't tell us the line of code causing an error – it points to the place in the code where the error occurs, but it requires our knowledge of the rest of the code to figure out where the line *causing* the error is.

3. Pesky Insects

The following examples demonstrate some bugs that people on the internet have run into. We're asking you, as 61A students, to help debug their code.

Note that the code below has been modified a bit, in order to preserve the anonymity of the original posters.

(a) Here is the erroneous code:

```
1  scores = []
2  num_tests = 10
3  for i in range(num_tests):
4      score = i * 2 + 28.4
5      scores.append(score)
6  passed = 0
7  for i in range(num_tests):
8      if any(scores > 40 for y in len(scores)):
9          passed = passed + 1
10 print(passed, "tests passed!")
```

The expected behavior is to print `4 tests passed!`. What we actually get is:

```
Traceback (most recent call last):
  File "/Users/internet/cs61a/tests.py", line 8, in <module>
    if any(scores > 40 for y in len(scores)):
TypeError: 'int' object is not iterable
```

i. Rewrite line 8 to loop over all of the scores in the scores list. You should *only* fix the `TypeError` in this question – this will still lead to erroneous output, but this is expected.

```
if any(y > 40 for y in scores):
```

ii. After rewriting the line above, we run the code again to see `10 tests passed!`, which is still incorrect. Rewrite line 8 *again* to correct this new issue.

```
if scores[i] > 40:
```

Note: This question is designed to emulate the general flow of thought when debugging a problem – usually, you want to start by fixing the immediate error in front of you, and then go step-by-step until your code works as intended. Of course, if you can see the entire error at once, then you're welcome to just fix it at once, but more often than not, it's hard to diagnose *all* the errors in a piece of code just by looking at one error message. It's usually much less time-consuming to go through things step-by-step until you figure out the way to make your code work as expected, which is what we demonstrate here.

(b) Here is the erroneous code:

```
1 percentages = [1, 0.9, 0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.2, 0.1, 0]
2 max_score = 300
3
4 scores = [[int(percent * max_score) for percent in percentages]
5
6 print("Scores corresponding to the given percentages, on a scale out of 300:")
7 print(scores)
```

The expected behavior is to print the following:

```
Scores corresponding to the given percentages, on a scale out of 300:
[300, 270, 240, 210, 180, 150, 120, 90, 60, 30, 0]
```

But the actual observed behavior is the following:

```
  File "/Users/internet/cs61a/scores.py", line 6
    print("Scores corresponding to the given percentages, on a scale out of 300:")
    ^
SyntaxError: invalid syntax
```

i. What line of code should you fix to prevent this error from happening?

4

ii. Rewrite the line you identified above to fix the error.

scores = [int(percent * max_score) for percent in percentages]

Note: Watch your brackets and parentheses!

4. Classes, Classes Everywhere

Now that we've done some practice with debugging, let's take it to the next step by incorporating some of the more recent concepts we've learned (specifically OOP)!

In each of the parts below, you'll see some code and observed result from running/using that code. Find the problem and fix it.

(a) Here is the code:

```
1 class Human:
2     def __init__(name, age):
3         self.name = name
4         self.age = age
5
6     def say(self, msg):
7         print(msg + ", said " + self.name)
```

Here is the observed behavior:

```
>>> bob = Human("Bob", 5)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: __init__() takes 2 positional arguments but 3 were given
```

i. What line contains the error?

> **2**

ii. Rewrite the line you identified above to fix the error.

> `def __init__(self, name, age):`

(b) In the question above, we defined a `Human` class. Now, we're going to define a subclass of this class called `Baby`, which will represent a specific type of human being (particularly a newborn). In this case, we only need to acquire the name of the baby from the user (the age of every baby upon creation is 0).

Here is the code:

```
1 class Baby:
2     def __init__(self, name):
3         super().__init__(name, 0)
4
5     def say(self, msg):
6         print("Wah!, cried " + self.name)
```

Here is the observed behavior:

```
>>> alice = Baby("Alice")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/Users/classes/cs61a/baby.py", line 3, in __init__
    super().__init__(name, 0)
TypeError: object.__init__() takes exactly one argument (the instance to initialize)
```

i. What line of code should you fix to prevent this error from happening?

> **1**

ii. Rewrite the line you identified above to fix the error.

> `class Baby(Human):`

Note: Watch your parameters! Remember that, almost always, the first parameter in a method in a class will be `self`. Remember also that this parameter is almost always implicit, especially when you access a method using `instance.method()` notation (as well as when you call `super()`).