[9 MARKS]

This question includes several small short-answer questions. Use any editor to type your answers, and save them in a file called Q1_answers.txt. Hand in this file on MarkUs.

Part (a) [4 MARKS]

Suppose we have an abstract class called `Monster` with two implemented methods (an initializer, and a method called `play`) and one abstract method called `learn`. Assume each method has one parameter, which is a string. (It doesn't matter to this question what the methods do.)

Suppose `Monster` has two subclasses: `Godzilla` and `Werewolf`.

1. Write 1-3 lines of client code that **clearly demonstrates polymorphism**.

   State the exact expression that is polymorphic in your code.

   ```
   >>> Godzilla.play("hi")
   "bye"
   >>> Werewolf.play("hi")
   "rawr!"
   ```

   Werewolf.play("hi") is polymorphic since the method changes what it does.

2. What method do we know for sure `Godzilla` must implement?

   learn

3. What is one reason why we might write a `play` method in class `Werewolf`? Be specific.

   the play method available is not suitable for it's docstring

Part (b) [3 MARKS]

The built-in `list` class has a method `reverse` that mutates the list, putting it in reverse order. The following hypothesis test will check one aspect of this method. Write the body of `test_reverse_count_unchanged`.

```
@given(lists(integers()), integers())
def test_reverse_count_unchanged(lst: List, item: int):
    """Test that reversing a list does not change the number of occurrences
    of an item."""
```

```
initial = lst.count(item)
reverse = lst.reverse()
assert initial == reverse.count(item)
```

Part (c) [2 MARKS]

Consider the following module:

```
from typing import List

def f3(lst: List[int]) -> float:
    n = 0
    total = 0
    for item in lst:
        if item % 2 == 0:
            total += item
            n += 1
    return total / n

def f2(lst: List[int]) -> float:
    return f3(lst)

def f1(lst: List[int]) -> float:
    try:
        return f2(lst)
    except ZeroDivisionError:
        print('Ouch!')
        return 0.0

if __name__ == '__main__':
    print(f1([3, 21, 85, 11, 7]))
```

1. How many stack frames are on the call stack, including the frame for `__main__`, when an error is raised?

   4

2. When the error is raised, how many stack frames are popped off the stack before "Ouch!" is printed?

   2