Single Layer of Abstraction Principle, CS 257 Name: _____

1. Write a TO paragraph for the following function (which is not at a single layer of abstraction):

```
def attack_with_dragon(name, color, size):
  """Attacks the target with a dragon of the given name, color, and size."""
  dragon = {
      "name": name,
      "color": color,
      "size": size,
  }
  if dragon["size"] == "large":
    print(f"The dragon {dragon['name']} breathes fire on the target\
        and burns it to a crisp!")
  else:
    print(f"The dragon {dragon['name']} breathes fire on the target\
        but it doesn't do any damage.")

  return dragon
```

TO attack with a dragon...

2. How could you refactor that function into multiple functions to get it to a single layer of abstraction?

3. Write a TO paragraph for the following function (which is not at a single layer of abstraction):

```
def get_user_info():
  """Gets the user information for a given username from a database"""

  #First login to the database
  username = input("Enter your username: ")
  password = input("Enter your password: ")
  connect_to_database(username, password)

  #Then get the information
  user_info = get_row_from_database(username)
  return user_info
```

TO get user information from the database...

4. How could you refactor that function into multiple functions to get it to a single layer of abstraction?

5. Write a TO paragraph for the following function (which is not at a single layer of abstraction):

```
def get_strongest_pokemon(types):
  """Gets the strongest Pokemon of the given types."""
  pokemon_data = get_pokemon_data()
  strongest_pokemon = None
  for pokemon in pokemon_data:
    if pokemon.types == types:
      pokemon.damage = pokemon.attack_power * 2 / pokemon.defense_power
      if strongest_pokemon is None or pokemon.damage > strongest_pokemon.damage:
        strongest_pokemon = pokemon
  return strongest_pokemon
```

TO get the strongest pokemon of a given type...

6. How could you refactor that function into multiple functions to get it to a single layer of abstraction?