Name: _____ Recitation: _____ Andrew Id: _____

15-112 Fall 2018 Quiz 9

Up to 20 minutes. No calculators, no notes, no books, no computers. Show your work!

1. (20 points) For each question, fill in the circle for **all** of the answers that are correct. **Some questions may have more than one correct answer.**

   (a) What is a Class in Python?
   - ◯ A specialized dictionary
   - ◯ A template
   - ◯ A specific item
   - ◯ A specialized function

   (b) What is an Instance in Python?
   - ◯ A type
   - ◯ A template
   - ◯ A specific item
   - ◯ The opposite of Assert

   (c) What is a Constructor in Python?
   - ◯ A method that makes an instance
   - ◯ The place where attributes are first set up
   - ◯ A generic object that makes an instance
   - ◯ A method called to create new classes

   (d) Which of the following can be a superclass of Guitar?
   - ◯ Music
   - ◯ StringInstrument
   - ◯ BassGuitar
   - ◯ Harp

   (e) What does it mean for a class to override a method?
   - ◯ It gets the method from its superclass
   - ◯ It gets the method from its subclass
   - ◯ It implements the method multiple times with new arguments
   - ◯ It changes how the method works from the original version

2. (25 points) **Free Response:** Write the function `reduceToStrings(lst)` which takes a list of values, `lst`, and returns a list containing all of the strings that occurred in `lst` in their original order of appearance. `lst` can contain non-string items (like ints or other lists). For example, `reduceToStrings([1, "ab", True, "car"])` would return `["ab", "car"]`. Strings inside nested lists should also be ignored; therefore, `reduceToString(["hello", ["what"], "world"])` should return `["hello", "world"]`.

   This function must be written recursively. A solution that uses loops, comprehensions, generators, or iterative built-in functions such as range will receive no credit.

3. (20 points) **Code Tracing:** Indicate what the following program prints. Place your answer (and nothing else) in the box to the right of the code.

```
def ct(s, depth=0):
    print(depth, "in:", s)
    if len(s) == 1:
        result = s
    elif s[0] in "aeiou":
        result = ct(s[1:], depth+1)
    else:
        result = s[0] + ct(s[1:], depth+1)
    print(depth, "out:", result)
    return result

ct("hope")
```

4. (35 points) **Free Response:** Write the classes Vehicle and Car so that they pass the following test cases. You may not hardcode any test cases. For full credit you must use inheritance appropriately.

```
# A Vehicle has one property: whether or not it is currently moving.
v1 = Vehicle(False)
assert(str(v1) == "Vehicle(False)")
# A vehicle can move and brake
v1.move()
assert(str(v1) == "Vehicle(True)")
v1.brake()
assert(str(v1) == "Vehicle(False)")
assert(str(Vehicle(True)) == "Vehicle(True)")

# A Car is a vehicle that has an engine. The engine must be on for the car to move
# Note that the first param is related to moving; the second checks the engine.
c1 = Car(False, False)
assert(str(c1) == "Car(False,False)") # list the moving state first
ok = False
try:    c1.move()
except: ok = True
assert(ok) # move() should crash if the engine isn't on
assert(str(c1) == "Car(False,False)")
c1.startEngine()
assert(str(c1) == "Car(False,True)")
c1.move()
assert(str(c1) == "Car(True,True)")
c1.brake()
assert(str(c1) == "Car(False,True)")
# Nothing stops us from making cars with weird start states
assert(str(Car(True, False)) == "Car(True,False)")

# Check for inheritance
assert(isinstance(c1, Vehicle) == True)
assert(isinstance(v1, Car) == False)
ok = False
try:    v1.startEngine()
except: ok = True
assert(ok) # Vehicles should not have engines
```

YOU MAY CONTINUE WRITING CODE ON THIS PAGE.