3) [20 points] Consider the following abstract class for a Car object.

class Car:
    """ Abstract class ""
    def __init__(self, max_speed, color):
        # max_speed is an integer and color is a string
        self.max_speed = max_speed
        self.color = color
    def get_max_speed(self):
        return self.max_speed
    def print_color(self):
        print "This car is " + self.color

a) [10 points] Define a subclass of Car named Convertible and fill in the boxes and the methods below.

class Convertible(Car):
    """ Convertible class inherits from Car ""
    def __init__(self, max_speed, color, roof_up):
        """ Initializes a car with max_speed (integer) and color (string). Adds a boolean data member for whether the roof is up or down ""

        Car.__init__(self, max_speed, color)
        self.roof_up = roof_up

    def get_roof_state(self):
        return self.roof_up

    def change_roof_state(self):
        """ Changes roof_up to the opposite of its current value ""

        self.roof_up = not self.get_roof_state()
def print_roof_state(self):
    """ prints "The top is up" if roof_up is True and
    "The top is down" when roof_up is False """
    if self.get_roof_state():
        print "The top is up"
    else:
        print "The top is down"

def get_max_speed(self):
    """ Returns the max_speed plus 50 """
    return 50 + self.max_speed

b) [10 points] What does the following code print out? If a line fails, write “error” for the
output of that line and go on to the next line.

```python
cl = Car(20,"blue")
print cl.get_max_speed()
c1.print_color()
c1.print_roof_state()

c2 = Convertible(20,"red",True)
c2.print_roof_state()
c2.change_roof_state()
c2.print_roof_state()
c2.print_color()
print c2.get_max_speed()
```

```
20
This car is blue
error
The top is up
The top is down
This car is red
70
```
4) [10 points] Consider the following sorting function:

```python
def swapSort(L):
    """ L is a list of integers """
    print "Original L: ", L
    for i in range(len(L)):
        for j in range(i+1, len(L)):
            if L[j] < L[i]:
                # the next line is a short
                # form for swap L[i] and L[j]
                L[j], L[i] = L[i], L[j]
    print L
    print "Final L: ", L
```

a) Does this algorithm sort in increasing or decreasing order?

Increasing

b) What is the worst case complexity of swapSort?

\(O(n^2)\)

c) If we modify the line: for j in range(i+1, len(L)): to be
for j in range(len(L)): what happens to swapSort ?

Sorts in decreasing order