Graded Exercise 3

CSC140 Foundations of Computer Science

04 March 2020

Questions and answers

1. In two or three sentences, describe a lesson that you drew from your reading of Tom Wolfe's *Robert Noyce and His Congregation*.

Robert Noyce is one of many pioneers of electrical and software engineering whose origins were in the midwestern and western parts of the United States. The culture of small towns, coming in part out of the churches that served those towns, cultivated independence, embrace of hard work, a practical bent, esteem for engineering, and a disregard for rigid formality and hierarchies. Noyce took those attitudes and habits with him to Silicon Valley, where he recreated some of the culture of a small town in Iowa in one of the world's most successful technology companies.

2. In two or three sentences, describe a lesson that you drew from your reading of Fred Brooks' *No Silver Bullet: Essences and Accidents of Software Engineering*.

Brooks predicts slow and incremental progress in software engineering. He argues that some aspects of creating software are unavoidably difficult and so rapid, revolutionary improvements are unlikely. We can improve our own software engineering skills more by learning how to divide a big problem into smaller problems, how to represent data, and how to reason

about the transformations of data than we can by concentrating on, for example, the rules of programming languages.

The next questions refer to the program that follows the questions.

---

3. What does this program accomplish in lines 49–51?

This part of the program creates two instances of the Time class and then computes and prints the sum of the two times.

4. What does this program accomplish in lines 55–62?

This part of the program creates instances of the Time class. Each instance contains a random number of hours and minutes. The code puts each instance in a list named appointments.

5. Write code that will create two instances of the Time class and assign the sum of the two Times to a third variable.

```
1       a = Time( 1, 50 )
2       b = Time( 1, 20 )
3       c = a.add(b)
```

6. Write code that will print all elements of a list that contains instances of the Time class.

```
1       for t in appointments:
2           print( t )
```

7. The selection_sort () function calls two other functions that the programmer has defined in this program. What are the names of these two functions?

The selection_sort () function calls pos_min() and swap() functions.

8. Execution of the statement on line 49 will cause the execution of the constructor of the Time class. On which lines does the program define that constructor?

The definition of the class' constructor is on lines 7–9.

9. Execution of the statement on line 65 will cause the execution of which method in the Time class?

It will cause the execution of the __str__() method.

10. The grammar of the Python programming language, unlike the grammars of many other programming languages, does not give programmers a means of creating a variable whose value cannot be changed. However, Python programmers commonly use a naming convention to indicate that they wish to use a variable as a constant.

    This program defines two constants. What are their names?

```
1 COUNT
2
3 MINUTES_IN_AN_HOUR
```

11. Show or explain how you could change the program so that it prints a Time as, for example, "1:20" instead of as "1 hour, 20 minutes."

```
1     def __str__(self):
2         return f"{self.hours:2d}:{self.minutes:2d}"
```

12. The program repeats the arithmetic in line 18 in lines 20 and 21. When experienced programmers see that they are repeating themselves, they write a function.

    Show or explain how you could define another method in the Time class. This new method will have no parameter other than self. It will return to its caller an integer. This integer will be the total number of minutes in the Time object.

    For example:

```
1     meeting = Time( 2, 45 )
2
3     # this next statement prints 165
4     # (because 2 hours and 45 minutes is
5     # 2 hours * 60 minutes/hour + 45 minutes = 165 minutes)
6     print( meeting.total_minutes() )
```

```
    def total_minutes(self):
        return self.hours * MINUTES_IN_AN_HOUR + self.minutes
```

Code: Defining a class and sorting a list.

```
import random

COUNT = 12
MINUTES_IN_AN_HOUR = 60

class Time:
    def __init__(self, hours, minutes):
        self.hours = hours + minutes // MINUTES_IN_AN_HOUR
        self.minutes = minutes % MINUTES_IN_AN_HOUR

    def add(self, other_time):
        hrs = self.hours + other_time.hours
        min = self.minutes + other_time.minutes

        return Time( hrs, min )

    def compare_to(self, other_time):
        my_minutes = self.hours * MINUTES_IN_AN_HOUR + self.minutes

        other_minutes = other_time.hours * MINUTES_IN_AN_HOUR
        other_minutes += other_time.minutes

        if my_minutes < other_minutes:
            return -1
        elif my_minutes == other_minutes:
            return 0
        else:
            return +1

    def __str__(self):
        return f"{self.hours:2d} hours, {self.minutes:2d} minutes"

```

```
33 def swap( appointments, i, j ):
34     appointments[i], appointments[j] = appointments[j], appointments[i]
35 
36 def pos_min( appointments, start_index ):
37     best_guess_so_far = start_index
38     for i in range(start_index + 1, len(appointments)):
39         if appointments[i].compare_to( appointments[best_guess_so_far] )
40             best_guess_so_far = i
41     return best_guess_so_far
42 
43 def selection_sort( appointments ):
44     for i in range(len(appointments)):
45         j = pos_min( appointments, i )
46         swap( appointments, i, j )
47 
48 if __name__ == "__main__":
49     a = Time( 1, 50 )
50     b = Time( 1, 20 )
51     print( f"{a} + {b} = {a.add(b)}" )
52     print( "a compared to  b?", a.compare_to(b) )
53     print( "b compared to  a?", b.compare_to(a) )
54 
55     appointments = []
56     for i in range(COUNT):
57         hrs = random.randint(0, 23)
58         min = random.randint(0, 59)
59 
60         t = Time( hrs, min )
61 
62         appointments.append( t )
63 
64     for t in appointments:
65         print( t )
66     print( "\n" )
67 
68     swap( appointments, 2, 4 )
69 
70     for t in appointments:
71         print( t )
72     print( "\n" )
73 
74     selection_sort( appointments )
75 
76     for t in appointments:
77         print( t )
78     print( "\n" )
```

```
79
80          print( "Hello" )
```