Student Name:	**Student ID:** **Section:** **Exam Room:**		**Total Grade:**
	Exam: Final	**Q1: 20**	**Q2: 20**
Acad. Year: 2018 - 2019 Fall	**Date: 08/01/2019**	**Q3: 20**	**Q4: 40**
Ins: Aynur Dayanık **İpek Sözen** **Lori Russell Dağ**	**Duration: 110 min.**		

Question 1 (20 pts.)

Write a python script that does the following. ***You should not use any loops in your solution***.

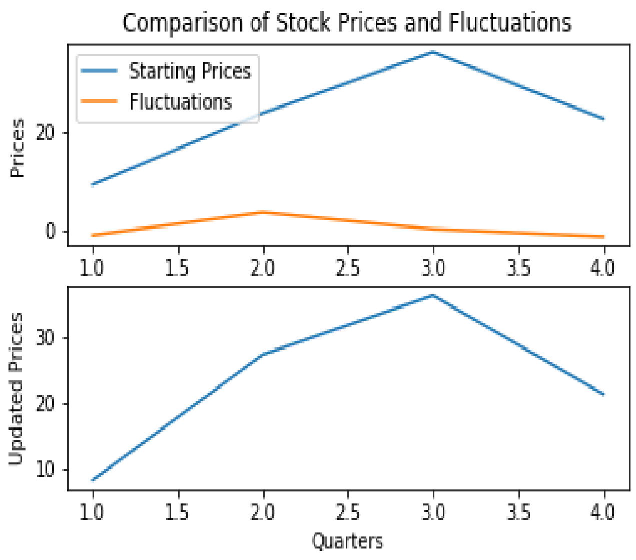
a. Using the appropriate functions, generate two numpy arrays, one that stores quarters (one to four) and second that stores the prices: [9.25, 23.6, 35.87, 22.5]
b. Prices fluctuate between -2 and 5 each quarter. Generate an array with random values to represent the fluctuations each quarter. **Hint:** numpy's random.rand(n) function returns n values between 0 and 1.
c. Create a new array that stores the updated prices (price + fluctuation).
d. Display the prices, fluctuations and the updated prices.
e. Using the arrays defined above, create the following plot. Pay attention to details such as labels,etc.

Sample Output:

```
Prices:  [ 9.25 23.6  35.87 22.5 ]

Fluctuations:  [0.73633045 1.94601354 3.50677188 2.23727131]

Updated Prices:  [ 9.98633045 25.54601354 39.37677188 24.73727131]
```



```
import numpy as np
import matplotlib.pyplot as plt


quarters = np.arange(1, 5)

s_prices = np.array([9.25, 23.6, 35.87,  22.5])

fluctuations = 7 * np.random.rand(4) - 2

update_prices = s_prices + fluctuations


print('Prices: ', s_prices)

print('Fluctuations: ', fluctuations)

print('Updated Prices: ' , update_prices)


plt.clf()


plt.subplot(2, 1, 1)

plt.plot(quarters, s_prices, quarters, fluctuations)

plt.title('Comparison of Stock Prices and Fluctuations')


plt.ylabel('Prices')

plt.legend(['Starting Prices', 'Fluctuations'])


plt.subplot(2, 1, 2)

plt.plot(quarters, update_prices)

plt.xlabel('Quarters')

plt.ylabel('Updated Prices')
```

Question 2 (20 pts.)

Trace the following and show the ***exact*** output(display) in the space provided

a. (12 pts.)

```
import numpy as np

arr1 = np.array([17, 22, 5, 120, 15, 72])

arr2 = arr1 > 50
print('Array 2:', arr2)

arr3 = arr1[arr2]
print('Array 3:', arr3)

arr4 = np.where(arr1 > 50)
print('Array 4:', arr4)

arr5 = np.linspace(0, 10, 6)
print('Array 5:', arr5)
```

```
Array 2: [False False False  True False  True]

Array 3: [120  72]

Array 4: (array([3, 5]),)

Array 5: [ 0.  2.  4.  6.  8. 10.]
```

b. (8 pts.)

```
class A:
    def fcn(self, x):
        return x * 3

class B(A):
    pass

class C(A):
    def fcn(self, x, y = 3):
        return x + y

class D(B):
    pass

objA = A()
objB = B()
objC = C()
objD = D()

print(objA.fcn( 5 ))
print(objB.fcn( 8 ))
print(objC.fcn( 2 ))
print(objD.fcn( 4 ))
```

```
15

24

5

12
```

Question 3 (20 pts.)

Write a function, max_chars() that takes a 2D list of strings as a parameter and returns a list containing the ***maximum*** length of the strings in **each** column.

Sample Data (`sentences` may change):

```
sentences = [['lion', 'house', 'car'],
              ['cat', 'dog', 'car'],
             ['cat', 'house', 'dragon']]

max_chars(sentences) returns [4, 5, 6]
```

```
def max_chars(table):
    max_count = []

    for col in range(len(table[0])):
        max_c = 0
        for row in range(len(table)):
            if len(table[row][col]) > max_c:
                max_c = len(table[row][col])
        max_count.append(max_c)
    return max_count
```

Question 4 (40 pts.)

(15 pts.) Create a Time class with the following data and methods. All attributes should be **private** and should be named appropriately.

a. __init__ method should initialize the hours and minutes to the values passed as parameters. **All data members should be private.**
b. Method addTime() which **updates** the Time (hours and minutes) with the minutes passed as an int parameter. E.g. (2 hour and 50 min) + (80 minutes) is (4 hr and 10 min)
c. Method __lt__() that compares two Time objects, and returns True if the time is earlier than the parameter time, False if not.
d. Method __repr__() which returns a string representation of the time in the following format: `08:40`

```
class Time:
    def __init__(self, hour, minute):
     self.__hour = hour
     self.__minute = minute


    def addTime(self, minutes):
       total_mins = self.__minute + minutes
       self.__minute= total_mins % 60
       hours = self.__hour + total_mins // 60
       self.__hour = hours % 24


    def __lt__(self, other):
        if self.__hour < other.__hour:
           return True
        elif self.__hour==other.__hour and self.__minute< other.__minute:
           return True
        return False


    def __repr__(self):
        return '{0:02d}:{1:02d}'.format(self.__hour,self.__minute)
```

(25 pts.) Create a python script that does the following:

a. Create an empty list named appointments, which will store appointment Times for a doctor.
b. Using data from the file, schedule_times.txt, store each Time object in the appointments list.
c. Sort the list of times.
d. The doctor may be delayed, and will have to postpone her/his appointments for the day from the morning (starting at 8:30) or the afternoon(starting at 12:30). Your script should input from the user when the delay will start, and the number of minutes to delay. You should then update all times in the list using the appropriate functionality.
e. Display the updated list of times.
f. Input a name of a Lab Test to search for, and using the Patient functionality, display the lab result value for each Patient in the list whose result for the input lab test is out of range.

`schedule_times.txt` (sample only, contents may change)

```
8:40
12:55
9:15
14:25
18:45
23:55
```

Sample Run:

```
When will delay start (M)orning / (A)fternoon: M

How many minutes will the doctor be late: 20
Scheduled appointments: [09:00, 09:35, 13:15, 14:45, 19:05, 00:15]

When will delay start (M)orning / (A)fternoon: A

How many minutes will the doctor be late: 45
Scheduled appointments: [08:40, 09:15, 13:40, 15:10, 19:30, 00:40]
```

```
from Time import *

# input the list of appointments from the input file
sFile = open('schedule_times.txt', 'r')
appointments = []

for time in sFile:
    h_m = time.split(':')
    hours = h_m[0]
    minutes = h_m
    appointments.append(Time(int(hours), int(minutes)))
sFile.close()




# sort the list of appointments
appointments.sort()



# input a time from the user
time = input('When will delay start (M)orning / (A)fternoon: ')

# input a number of minutes
mins = int(input('How many minutes will the doctor be late: '))


# shift each appointment according to the inputs
for a in appointments:
     if time.lower() == 'M'.lower():
          a.addTime(mins)
     elif time.lower() == 'A'.lower():
          #if not (a < Time(12,30)):
               #a.addTime(mins)
          +if a > Time(12,30):
               a.addTime(mins)

#print the scheduled appointments
print('Scheduled appointments:', appointments)
```