Practice Midterm A (Solutions)

Q1. Probability (10 points)

For the following question, you are not expected to compute the numerical answer, but should leave your answer either as mathematical expressions that can be easily calculated, or as Python code that can be executed. For clarity, you should draw a box around the expression containing your final answer.

A life insurance company would like to conduct an analysis on its policy holders. The database shows that 60% of policyholders are male, and 40% are female. In this population, assume that the lifespan of a man obeys a Normal distribution with mean 75 years and standard deviation 15 years. (Lifespan is defined as the number of years from birth to death.) The lifespan of a woman obeys a Normal distribution with mean 80 and standard deviation 12 years. All individuals' lifespans are independent of one another.

A. (3 points) What is the probability that a randomly drawn individual lives at least 85 years?

Let M denote the event that the policyholder is male, and W the reverse. Let X be a random variable denoting the policy holder's life span. Let F_M denote the CDF of a $Normal(75, 15)$ distribution and F_W denote the CDF of a $Normal(80, 12)$ distribution. We have

$$P(X \geq 85|M) = 1 - F_M(85),$$
$$P(X \geq 85|F) = 1 - F_W(85).$$

Hence, the desired probability is

$$P(X \geq 85) = P(M)P(X \geq 85|M) + P(F)P(X \geq 85|F) = 0.6(1 - F_M(85)) + 0.4(1 - F_W(85)).$$

B. (4 points) Given that a randomly drawn individual lives at least 85 years, what is the probability that the individual is male?

Using Bayes' rule (or equivalently by solving the joint probability table), the desired probability is

$$P(M|X \geq 85) = \frac{P(X \geq 85|M)P(M)}{P(X \geq 85)} = \frac{0.6(1 - F_M(85))}{0.6(1 - F_M(85)) + 0.4(1 - F_W(85))}.$$

C. (3 points) Given that a randomly drawn individual lives at least 85 years, what is the probability that the individual lives at least 5 more years?

By definition,

$$P(X \geq 90|X \geq 85) = \frac{P(\{X \geq 90\} \text{ and } \{X \geq 85\})}{P(X \geq 85)} = \frac{P(X \geq 90)}{P(X \geq 85)}.$$

Hence, using the formula from part A, the desired probability is,

$$P(X \geq 90|X \geq 85) = \frac{0.6(1 - F_M(90)) + 0.4(1 - F_W(90))}{0.6(1 - F_M(85)) + 0.4(1 - F_W(85))}.$$

Q2. Generating Simulated Data using Python (10 points)

A logistic company would like to simulate demand for a given product. Assume that there are Good or Bad weeks. On a Good week, the demand is Normally distributed with mean 200 and standard deviation 50. On a Bad week, demand is Normally distributed with mean 100 and standard deviation 30. (Assume that demand can take any decimal number, so you do not have to round to an integer. However, if the demand is ever negative, you should set it to be zero.)

Whether a week is Good or Bad is not independent across time. Conditional on a given week being Good, the next week remains Good with probability 0.8. Similarly, conditional on a given week being Bad, the next week remains Bad with probability 0.8.

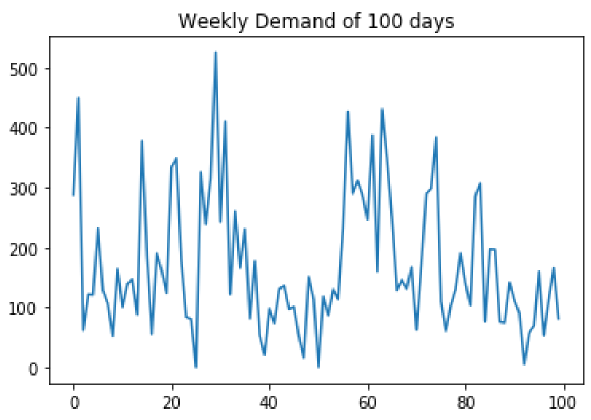
In the spaces below, write complete and executable Python code that creates a list called "demand" containing simulated demand for 100 weeks, assuming that the first week is Good. (The list should have 100 entries and the first entry is a float number giving the demand on the first week, the second entry for the second week, etc.) You must import all packages you use.

```
[1]: from scipy.stats import norm
     import numpy as np
     demand=[]
     market='good'
     distGood=norm(300,100)
     distBad=norm(100,50)
     for i in range(100):
         if market=='good':
             sample=distGood.rvs()
         else:
             sample=distBad.rvs()
         sample=max(0,sample)
         demand.append(sample)
         if market=='good':
             market=np.random.choice(['good','bad'],p=[0.8,0.2])
         else:
             market=np.random.choice(['good','bad'],p=[0.2,0.8])
```

The following code plots the output.

```
[5]: import pandas as pd
     pd.Series(demand).plot(title='Weekly Demand of 100 days')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f44c3cec630>
```



Q3. Analyzing Data using Python (10 points)

This question asks you to analyze the performance of a certain inventory replenishment policy, given a list of simulated weekly demand. The policy has two parameters, s and q. At the end of every week, if the remaining inventory is less than or equal to s, then the policy will order q units, which will arrive at the start of next week before next week's demand is incurred.

Assume that at the beginning of the first week, the inventory level is $s + q$. In any week, if you do not have enough inventory to fulfill demand, the excess demand you could not fulfill is lost.

In the spaces below, write a function called "analyze" with the following three input arguments:

- `demand`: a non-empty list of non-negative numbers, with each number representing the demand for a week. (This list is analogous to the output of Q2 but the length may not necessarily be 100.)
- `s`: the threshold at or below which you will place an order of q units.
- `q`: the amount to order each time the ending inventory in a week is below s.

The function should return three numbers:

- `avFulfilled`: the average demand per week that was successfully met by your inventory.
- `avLost`: the average demand per week that was lost (because you did not have enough inventory that week).
- `avPEI`: the average inventory at the end of each week at the time of ordering (but before the order arrives).

Note: the syntax for returning multiple values, say `A`, `B` and `C`, is as follows:

```
return A, B, C
```

Example:

```
analyze([10,30,80,30,80],30,40)
```

yields the result
`38.0, 8.0, 16.0`
The following table summarizes what is going on each week:

Week	Starting Inventory	Demand	Fulfilled	Lost	Ending Inventory	Order?
0	70	10	10	0	60	No
1	60	30	30	0	30	Yes
2	70	80	70	10	0	Yes
3	40	30	30	0	10	Yes
4	50	80	50	30	0	Yes
Total	–	–	**190**	**40**	**100**	–
Average	–	–	**190/5=38**	**40/5=8**	**100/5=20**	–

```
[3]: def analyze(demand,s,q):
         inventory=s+q
         totFulfilled=0
         totLost=0
         totPEI=0
         for curDemand in demand:
             fulfilled=min(curDemand,inventory)
             lost=curDemand-fulfilled
             inventory-=fulfilled

             totFulfilled+=fulfilled
```

```
            totPEI+=inventory
            totLost+=lost

            if inventory<=s:
                inventory+=q
        n=len(demand)
        return totFulfilled/n, totLost/n, totPEI/n
```

```
[4]: analyze([10,30,80,30,80],30,40)
```

```
(38.0, 8.0, 20.0)
```