Session 5: Algorithmic Thinking II

1) Describe the task succintly and precisely.
2) Decompose the task into components and describe how to do each in English.
3) Translate each component into code and test them independently.
4) Combine together and test.

Paper Coding Exercise (20 minutes)

Without using a computer, hand write Python code that implements the following logic: Given a list named `curVal`, representing the valuation (willingness to pay) of a customer for two products, as well as list named `priceVector`, representing the price of the two products, print `"Purchase product 0"` if the customer purchases the first product; print `"Purchase product 1"` if the customer purchases the second product; print `"Purchase nothing"` if the customer purchases neither.

Assumptions: If the customer's valuations for both products are greater than or equal to the corresponding prices, then the customer will purchase the product in which his/her valuation minus the price is the largest. If there is a tie, then the customer will purchase the first product.

For example, if the valuation of a customer is `[9,8]` then

- If `priceVector=[6,4]`, then the customer will purchase the second product because $8-4>9-6$.
- If `priceVector=[5,4]`, then the customer will purchase the first product because $9-5\geq 8-4$.
- If `priceVector=[10,8]`, then the constumer will purchase the second product.
- If `priceVector=[10,10]`, then the customer will purchase neither products.

```
[1]: # Input
     curVal=[25,15]
     priceVector=[25,10]
     # Write your code below
```

After you are done, trace through your code several times with different values of `curVal` and `priceVector` and check for syntax or logical errors.

Q2: Exchange your code with a neighbor and help one another check for errors. If you find an error, explain it to your neighbor with concrete inputs.

(Optional exercise if you finish early): Modify your code to work when `curVal` and `priceVector` are lists of arbitrary length. (Still do this on a piece of paper without the help of a computer.)

Case 8: Demand Estimation for Substitutable Products

Write a function named `demand` with two input arguments:

- `priceVector`: a list of length 2 containing two positive numbers, corresponding to the proposed prices for the two products.
- `values`: a list in which each element is a list of length 2, corresponding to the valuation of a customer for the two products.

The function should return a list of two numbers, representing the number of customers purchasing each product. Assume the same customer behavior as in the paper coding exercise.

```
values=[[25,15],[18,18],[30,20],[30,30]]
priceVector=[25,20]
demand(priceVector,values)
```

```
[2, 1]
```

I. Describe:

For each customer, figure out which of the products will he/she purchase, if any. Keep track of the total number of customers purchasing each product.

II. Decompose:

A. Loop through the customers.

B. Figure out which product will a given customer purchase, if any. (Paper coding exercise)

C. Keep track of the total number of customers purchasing each product: define a variable for each product tracking the number of customers purchasing that product so far, and incrementing this by one when needed.

III. Translate:

```
[2]: # A. Loop through...
     values=[[25,15],[18,18],[30,20],[30,30]]
     # Write your code below
```

```
25 15
18 18
30 20
30 30
```

```
[3]: # B. Figure out... (Paper coding exercise solution)
     curVal=[25,15]
     priceVector=[25,10]
```

```
Purchase product 1
```

[5]:
```
# C. Keep track...
count=[0,0]
# Write your code below



print(count)
```

```
[2, 1]
```

IV. Combine

[1]:
```
# Intermediate version with print outputs and no function encapsulation
values=[[25,15],[18,18],[30,20],[30,30]]
priceVector=[25,20]
```

```
Current value vector: [25, 15]
        Difference of valuation and prices 0 -5
        Purchase product 0
        Count: [1, 0]
Current value vector: [18, 18]
        Difference of valuation and prices -7 -2
        Purchase nothing
Current value vector: [30, 20]
        Difference of valuation and prices 5 0
        Purchase product 0
        Count: [2, 0]
Current value vector: [30, 30]
        Difference of valuation and prices 5 10
        Purchase product 1
        Count: [2, 1]
```

[7]:
```
# Code to test your final solution (after you encapsulate it in a function)
values=[[25,15],[18,18],[30,20],[30,30]]
priceVector=[25,20]
demand(priceVector,values)
```

```
[2, 1]
```

Case 9. Queuing Analysis

A popular fast food restaurant is planning to open a branch in a new location and wants to decide how many servers to hire. To help them, write a function `queueLength` with two input parameters:

- `k`: the number of customers that can be served in a minute. (assumed to be integer)
- `demand`: a list of integers specifying how many customers arrive each minute. (For simplicity, assume that customers arrive at the beginning of each minute and up to k customers can be served instantly.)

Let T be the length of the list. The function should return the average queue length at the end of the given T minutes.

For example, if `k=3` and `demand=[2,3,6,8,10,2,1,0,1,0]`, the following table summarizes the evolution of the queue.

Minute	# of Arrivals	# Served	Queue Length at End of Minute
0	–	–	0
1	2	2	0
2	3	3	0
3	6	3	3
4	8	3	8
5	10	3	15
6	2	3	14
7	1	3	12
8	0	3	9
9	1	3	7
10	0	3	4
Average	3.3	–	**7.2**

To illustrate why this function is useful, according to a mathematical result known as Little's Law, the average queuing time of customers is

$$\frac{\text{Average Queue Length}}{\text{Average Arrival Rate}} = \frac{7.2}{3.3} \approx 2.2 \text{ minutes.}$$

I: Describe

II: Decompose

III-IV: Translate and Combine

Having this function allows the company to run the following analysis:

```
[9]: k=3
     demand=[2,3,6,8,10,2,1,0,1,0]
     print(f'Average queue length is {queueLength(k,demand)} customers.')
```

```
Average queue length is 7.2 customers.
```

```
[10]: import numpy as np
      print(f'Average queuing time is {queueLength(k,demand)/np.average(demand):.1f} minutes.
```

```
Average queuing time is 2.2 minutes.
```

```
[11]: # Find the k needed to keep average waiting time at or below 1.5 minutes.
      demand=[2,3,6,8,10,2,1,0,1,0]
      k=1
      while (queueLength(k,demand)/np.average(demand)>1.5):
          k+=1
      print(f'Service rate needed to keep waiting time below 1.5 minutes: k={k}.')
```

```
Service rate needed to keep waiting time below 1.5 minutes: k=4.
```