Department of Information Systems and Technologies

CTIS152 – Algorithms and Data Structures

FALL 2024 - 2025

Lab Guide #18 – Week 12 – 1

OBJECTIVE : Queue Operations, Stack&Queue Mix Questions
Instructor : Burcu LIMAN **Assistants :** Berk ÖNDER - Engin Zafer KIRAÇBEDEL

Q1. In a petrol station, daily sale information is kept in the file **customers.txt** such that each customer's information contains the **car plate, fuel type,** and **amount of petrol in liter bought** from this station.

Write a C program that gets several customers' information from the file and stores them in a "structure queue". Then the program displays a daily report.

Make the necessary changes in the header file to create a **structure** queue. (**carPlate, fuelType, liter, payment**)

Write the following functions;

- **`calculatePayment:`** gets customer information as a parameter, and calculates the payment according to the fuel type of the car. If the fuel is Diesel; the price is **`43.44`** TL per liter. On the other hand, if the fuel type is Fuel; the price is **`42.71`** TL per liter.
- **`fillQueue:`** reads the customer information from the file calculates the payment for each customer using the function above and inserts the customers into the queue.
- **`dailyReport:`** gets the customer queue as a parameter and displays a detailed daily report;
  - information of each customer including the payment,
  - the total payment and the total liter amount sold from the station.

Project Name: LG18_Q1
File Name: Q1.cpp

customers.txt

```
06UG215 F 90.55
06KF121 D 40.11
06SS350 F 80.20
06NV834 D 70.80
06KG507 F 25.14
06BV3911 F 43.23
06BC229 D 16.75
```

Example Run:

```
Plate      FuelType  Liter  Payment

*********  ********  *****  **********

06UG215        F     90.55  3867.39 TL

06KF121        D     40.11  1742.38 TL

06SS350        F     80.20  3425.34 TL

06NV834        D     70.80  3075.55 TL

06KG507        F     25.14  1073.73 TL

06BV3911       F     43.23  1846.35 TL

06BC229        D     16.75   727.62 TL


Total liter sold: 366.78

Total payment   : 15758.37 TL
```

Q2. Write a C program that reads information about several products **(id, price,** and **stock amount**) from a text file named **"items.txt"** and decides to store the product in a queue or a stack according to its stock amount. If a product's stock amount is **less than 100**, it will be added to the queue, otherwise to stack. Then, the contents of the queue and the stack will be displayed.

After that, for every item in the queue, ask the user to get a purchase amount. Validate the input so that the purchase amount added to the stock amount must be greater than or equal to 100. Then, add the purchase amount to the product's stock amount and insert the product into the stack. Display stack once more at the end of the program.

Write the following functions;

- **`displayQueue`:** displays the content of the queue.
- **`displayStack`:** that displays the content of the stack.

Project Name: LG18_Q2
File Name: Q2.cpp

Example Run:

```
Queue contents:
111  7.50   60
444  40.50  80
777  1.00   85
888  25.55  54

Stack contents:
999  24.00  189
666  21.00  167
555  32.60  410
333  9.50   246
222  9.35   530

Enter the purchase amount for Item 111 (min 40): 20
Wrong Input!
Enter the purchase amount for Item 111 (min 40): 55
Enter the purchase amount for Item 444 (min 20): 30
Enter the purchase amount for Item 777 (min 15): 8
Wrong Input!
Enter the purchase amount for Item 777 (min 15): 23
Enter the purchase amount for Item 888 (min 46): 60

Stack contents after purchases:
888  25.55  114
777  1.00   108
444  40.50  110
111  7.50   115
999  24.00  189
666  21.00  167
555  32.60  410
333  9.50   246
222  9.35   530
```

items.txt

```
111   7.50   60
222   9.35  530
333   9.50  246
444  40.50   80
555  32.60  410
666  21.00  167
777   1.00   85
888  25.55   54
999  24.00  189
```

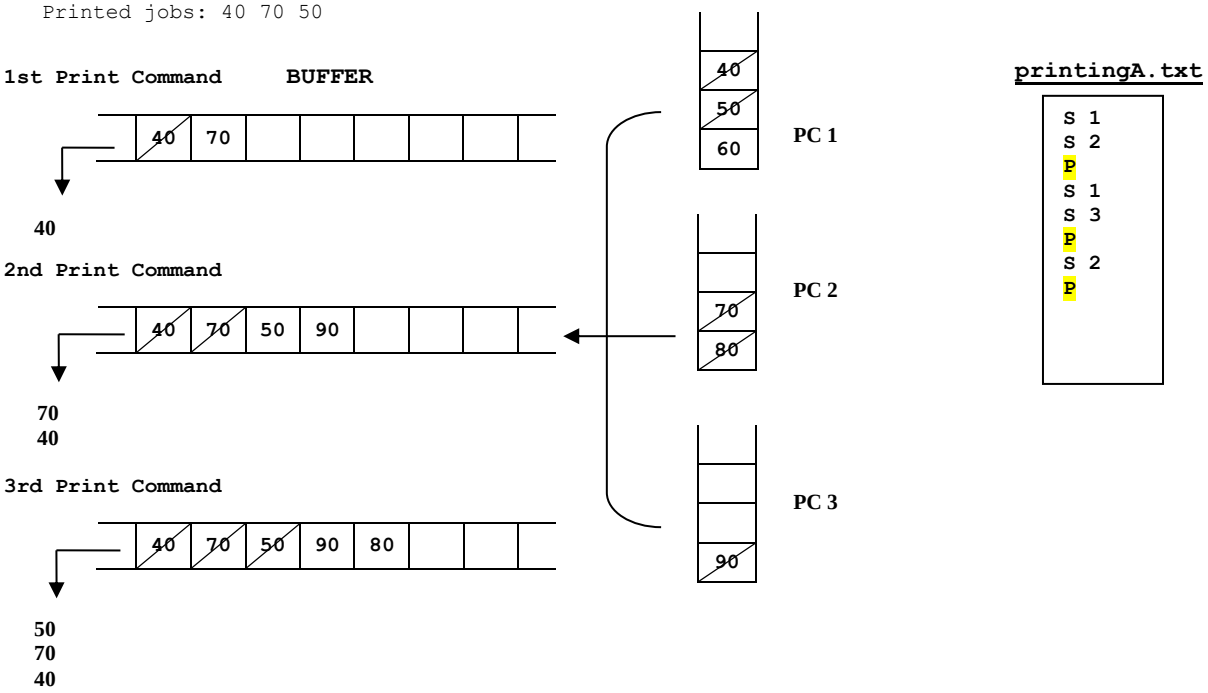
Q3. Write a C program that simulates a print server in the following way:

- There are **3 PCs** connected to a single printer. Print jobs are stored in a stack for each PC.
- Jobs are 'S'ent to the printer and since the printer can print one job at a time they start waiting in the printer buffer. The jobs are printed on a first-come-first-served basis and the printed job is removed from the buffer.
- Printer's buffer is represented by a queue (of **size 7**).
- Your program should perform this simulation in the following way:
- Initialize the stacks (# of the stack is 3) of the PCs to the following data (job-id's):
    - PC1 : {60, 50, 40}
    - PC2 : {80, 70 }
    - PC3 : {90 }
- From a text file called printingA.txt, read (**S**)end and (**P**)rint data (file contents are shown below).
- When a (**S**)end command is read, the job residing at the top of the specified stack is sent to the queue. e.g. "**S 2**" means send the available job from stack 2 to the printer queue.
- When a (**P**)rint command is read, the job at the front of the queue is printed (removed).
- At the end of the program (when the end of the file is reached) display pending jobs (still waiting at the printer buffer) and printed jobs. (Check the example run)

Project Name: LG18_Q3
File Name: Q3.cpp

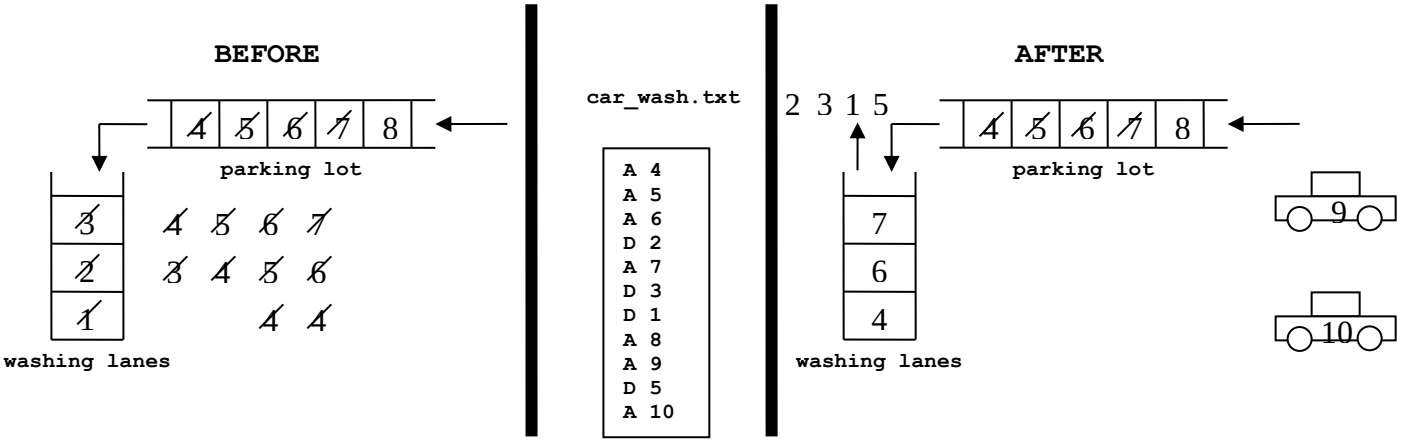
Example Run:

```
Pending jobs: 90 80
Printed jobs: 40 70 50
```



ADDITIONAL QUESTION

AQ. A carwash company has 3 washing lanes and a small parking lot for 5 cars. The washing lanes has a single entrance where only one car can pass at a time. When a car arrives **('A')** it passes through the parking lot and enters the washing lanes if there is an empty lane. ***At the beginning, assume the lanes are full (cars 1, 2 and 3 are being washed)***. When a car is washed and wants to depart **('D')** that car has to be moved out of the lane. ***If there are cars which have entered the lanes after this car, they have to be moved out first*** (and of course moved in back to continue to be washed). Meanwhile, remember that the parking lot (waiting cars) has a capacity as well, therefore, ***if the capacity is reached and new cars are arriving to be washed, they will not be allowed to enter the parking lot***.



Write the following function:

- **carWash** that gets stack, queue , char code and id of the car getting from the file, then makes the appropriate operation inside the stack and queue.

Write a C program that will read the information of the car from your text file and use the function above and print the queue after each operation.

Example Run:

```
The washing lanes:
3
2
1
The car Queue: 4
The washing lanes:
3
2
1
The car Queue: 4 5
The washing lanes:
3
2
1
The car Queue: 4 5 6
The washing lanes:
4
3
1
The car Queue: 5 6
The washing lanes:
4
3
1
The car Queue: 5 6 7
The washing lanes:
5
4
1
The car Queue: 6 7
The washing lanes:
6
5
4
The car Queue: 7
The washing lanes:
6
5
4
The car Queue: 7 8
The washing lanes:
6
5
4
The car Queue: 7 8 9
The washing lanes:
7
6
4
The car Queue: 8 9
The washing lanes:
7
6
4
The car Queue: 8 9 10
```