OS2 – Semaphore report

Student: Ahmed Al-Ganad – 528820

This document is the report for the Semaphore assignment. It covers the:

- Problem approach.
- Flowchart and brief explanation of the software.
- Software Tests with screenshots.
- Finally, a conclusion and reflection.

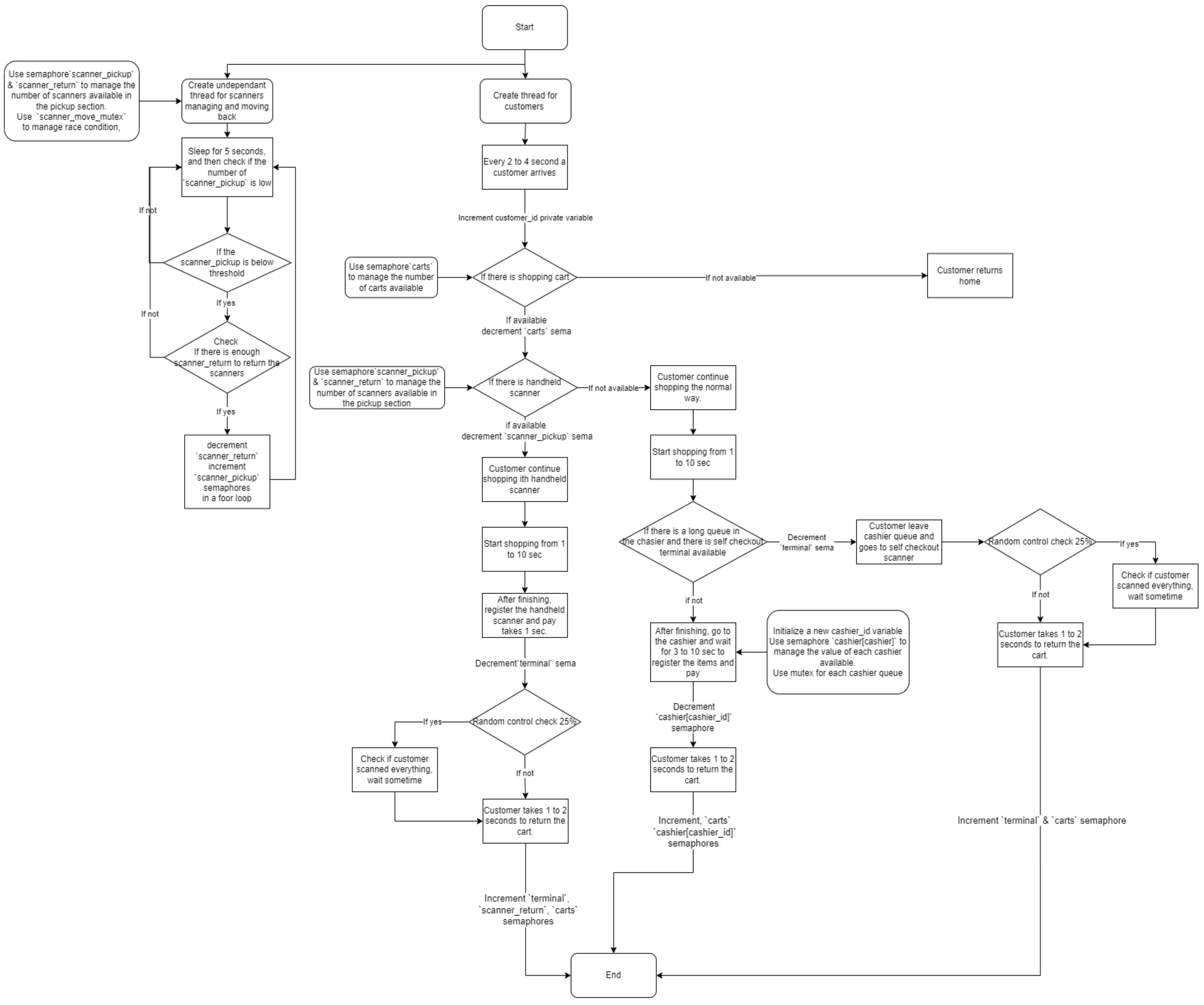
Problem approach:

After finishing the assignments of OS2, studying the slides, looking at a lot of videos in the internet about semaphores, and reading the documentation of important libraries such as semaphore.h semaphore.h(0p) - Linux manual page (man7.org), and pthread.h pthreads(7) - Linux manual page (man7.org) to help understand how to implement the software and to utilize useful functions.

This approach helped me understand the concept behind the topic and to implement the software of the semaphore assignment.

A flowchart will be provided in the next section to explain the software including semaphores, mutexes, and threads used. After that explanation of the semaphores, mutexes, and functions implemented will be present.

Schematics to describe the software:



Start
Use semaphore`scanner_pickup` & `scanner_return` to manage the number of scanners available in the pickup section. Use `scanner_move_mutex` to manage race condition,
Create undependant thread for scanners managing and moving back
Sleep for 5 seconds, and then check if the number of `scanner_pickup` is low
If not
If the scanner_pickup is below threshold
If yes
If not
Check If there is enough scanner_return to return the scanners
If yes
decrement `scanner_return` increment `scanner_pickup` semaphores in a foor loop
Create thread for customers
Every 2 to 4 second a customer arrives
Increment customer_id private variable
Use semaphore`carts` to manage the number of carts available
If there is shopping cart
If not available
Customer returns home
If available decrement `carts` sema
Use semaphore`scanner_pickup` & `scanner_return` to manage the number of scanners available in the pickup section
If there is handheld scanner
If not available
Customer continue shopping the normal way.
if available decrement `scanner_pickup` sema
Customer continue shopping ith handheld scanner
Start shopping from 1 to 10 sec
Start shopping from 1 to 10 sec
After finishing, register the handheld scanner and pay takes 1 sec.
Decrement`terminal` sema
Random control check 25%
If yes
Check if customer scanned everything, wait sometime
If not
Customer takes 1 to 2 seconds to return the cart.
Increment `terminal`, `scanner_return`, `carts` semaphores
If there is a long queue in the chasier and there is self checkout terminal available
Decrement `terminal` sema
Customer leave cashier queue and goes to self checkout scanner
Random control check 25%
If yes
Check if customer scanned everything, wait sometime
If not
Customer takes 1 to 2 seconds to return the cart.
if not
After finishing, go to the cashier and wait for 3 to 10 sec to register the items and pay
Initialize a new cashier_id variable Use semaphore `cashier[cashier]` to manage the value of each cashier available. Use mutex for each cashier queue
Decrement `cashier[cashier_id]` semaphore
Customer takes 1 to 2 seconds to return the cart.
Increment, `carts` `cashier[cashier_id]` semaphores
Increment `terminal` & `carts` semaphore
End

Figure 1- Supermarket simulation using threads, semaphores, and threads. Software Flowchart.

Description of the semaphores used:

- **`carts`:** to manage the carts available.
- **`terminal`:** to manage the terminals available.
- **`cashiers[CASHIERS]`:** to manage each cashier available.
- **`scanner_pickup`:** to manage the value of the scanners available in the pickup section.
- **`scanner_return`:** to manage the value of the scanners in the return section.

Description of the mutexes used:

- **` queue_mutex`:** mutex to protect racing condition on the section where the lengths of all cashier queues are checked and updated. It ensures that no other customer can interfere while a customer is finding the shortest queue and updating the length of the selected queue.
- **`cashier_queue[CASHIERS]` :** mutex to protect the racing condition for each cashier queue. It ensures that every customer can update the length of a certain cashier queue at a certain time.
- **`  scanner_move_mutex`:** mutex to protect from racing condition to return the scanners on the return section to the pickup section,

Description of the functions used:

- **void *handle_customers_arrival(void* arg)** : Function to handle the customer arrival, it creates a thread `customer_thread` and join to the function handle_carts()
- void *handle_carts(void* arg) : Function to handle the carts availability using a semaphore `cashier`, it creates a thread `customer_choice_thread` and joins it with the following function handle_customer_choice()
- **void *handle_customer_choice(void* arg):** Function to handle the customer choice between taking a handheld scanner for shopping the normal way, it uses the scanner_pickup semaphore, and it create a thread either to handle_self_checkout_shopping() func or handle_cashiers() func depending on the availability of the scanner_pickup semaphore.
- **void *handle_self_checkout_shopping(void* arg):** Function for people with handheld scanner to manage the shopping time and the self-checkout in the terminal, that uses `terminal` semaphoreit also has a random check for scanning all items it also posts or increment the `scanner_return` semaphore, the `terminal` semaphore, and the `carts` semaphore after the customer finish shopping.
- **void *handle_cashiers(void* arg):** Function to handle the cashiers queues, it uses 3 semaphores for 3 different cashiers `cashier[CASHIERS]` and uses a mutex `queue_mutex` to prevent the race condition between customers(threads), it also uses a mutex for each cashier `cashier_queue[CASHIERS]` it checks the condition if the cahier queue is long and there is a terminal free if so it creates a thread `self_checkout_thread` create a thread with handle_self_checkout_for_people_without_scanner() func so that the user without hand held can skip the cashier queue and go to the self-checkout terminal. If not, it posts `cashier[cashier_id]` and the `carts` semaphores.
- **void *handle_self_checkout_for_people_without_scanner(void* arg):** Function to handle people who skipped the cashier queue . This function is very similar to the handle_self_checkout_shopping() function but without posting the `scanners` semaphore.
- **void *move_scanners(void *arg):** Function that is responsible for returning the scanners to the pickup place it's thread is created and initialized in the main function so that it checks if there are scanners independently every 5 sec if there is low amount of scanners in the pickup section and enough scanners in the return section it waits and

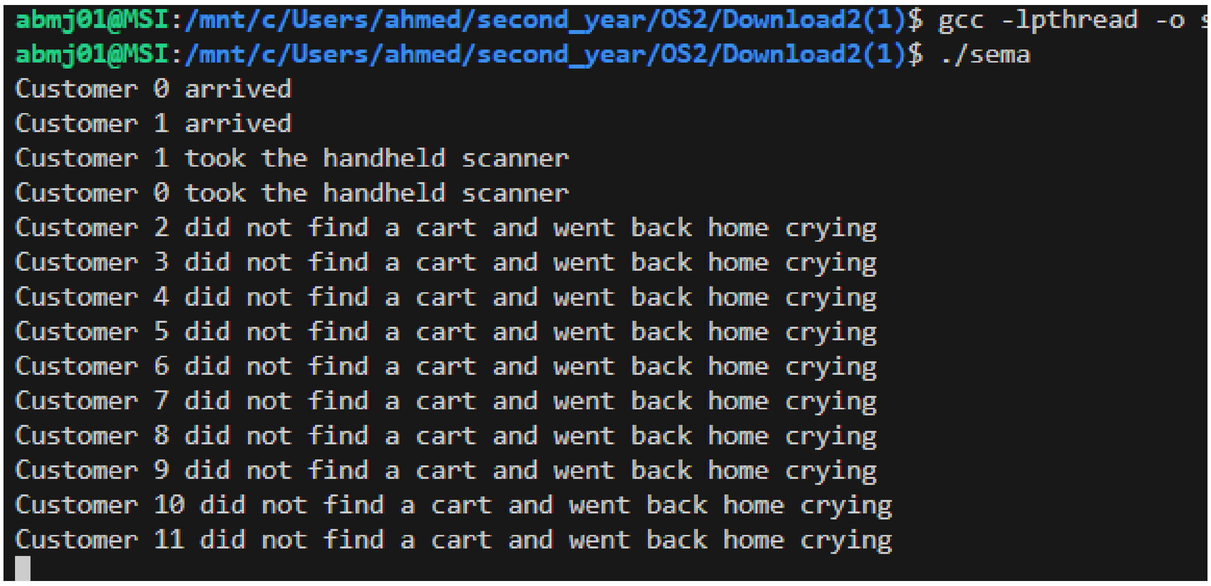
then decrement the value of the `scanner_return` and post or increment the value of the `scanner_pickup` in a for loop at once.

- **void wait_time(int min, int max):** Function to sleep() between min and max parameters.

Test conditions:

1- Customer leaves if there are not enough carts:

To test this: Decrease the number of carts to check:

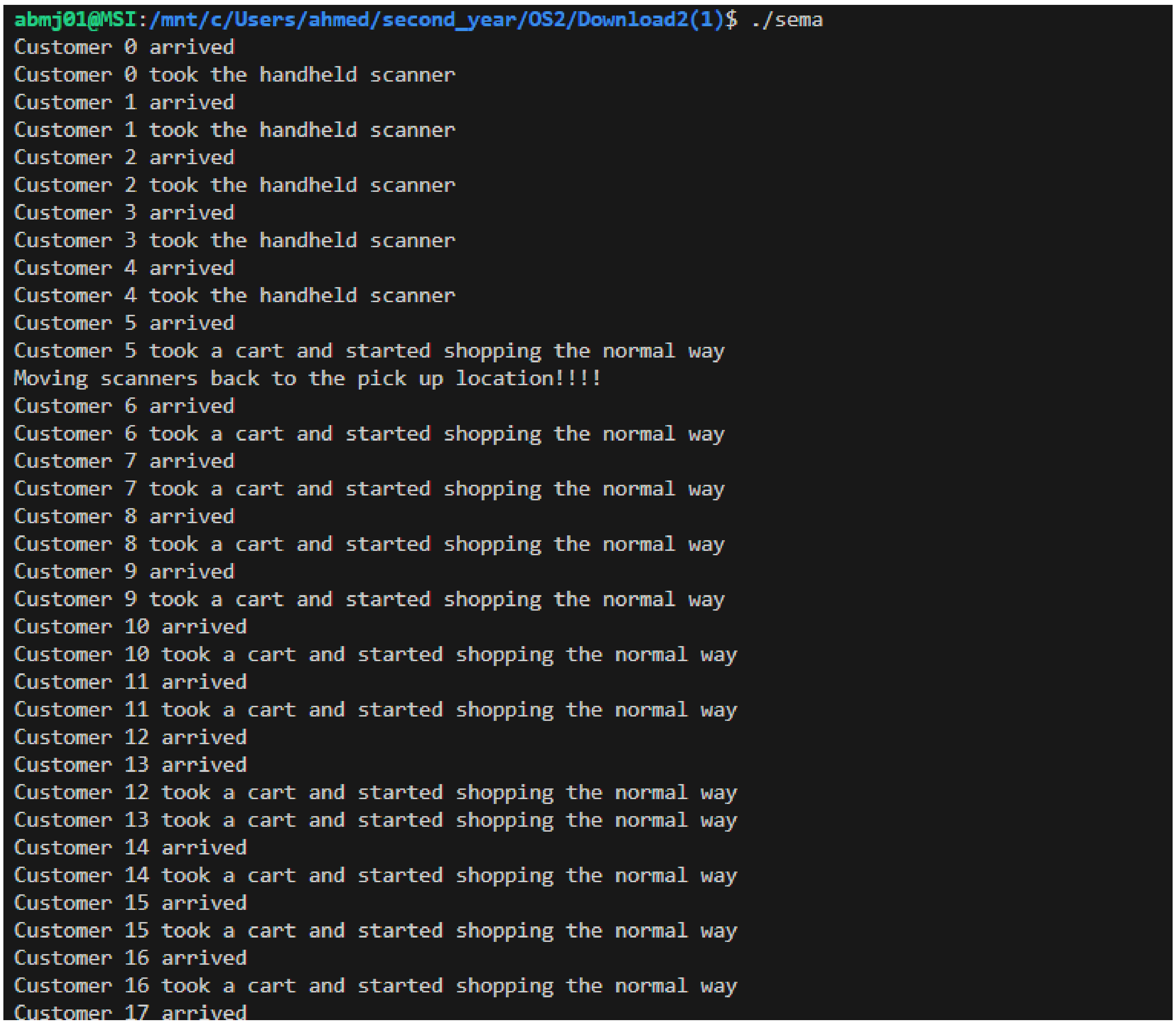


```
abmj01@MSI:/mnt/c/Users/ahmed/second_year/OS2/Download2(1)$ gcc -lpthread -o s
abmj01@MSI:/mnt/c/Users/ahmed/second_year/OS2/Download2(1)$ ./sema
Customer 0 arrived
Customer 1 arrived
Customer 1 took the handheld scanner
Customer 0 took the handheld scanner
Customer 2 did not find a cart and went back home crying
Customer 3 did not find a cart and went back home crying
Customer 4 did not find a cart and went back home crying
Customer 5 did not find a cart and went back home crying
Customer 6 did not find a cart and went back home crying
Customer 7 did not find a cart and went back home crying
Customer 8 did not find a cart and went back home crying
Customer 9 did not find a cart and went back home crying
Customer 10 did not find a cart and went back home crying
Customer 11 did not find a cart and went back home crying
```

Figure 2 - customer leaves no enough carts.

2- Customers choose between scanner if available or normal shopping:

To test this: Decrease the number of scanners and increase the number of carts:

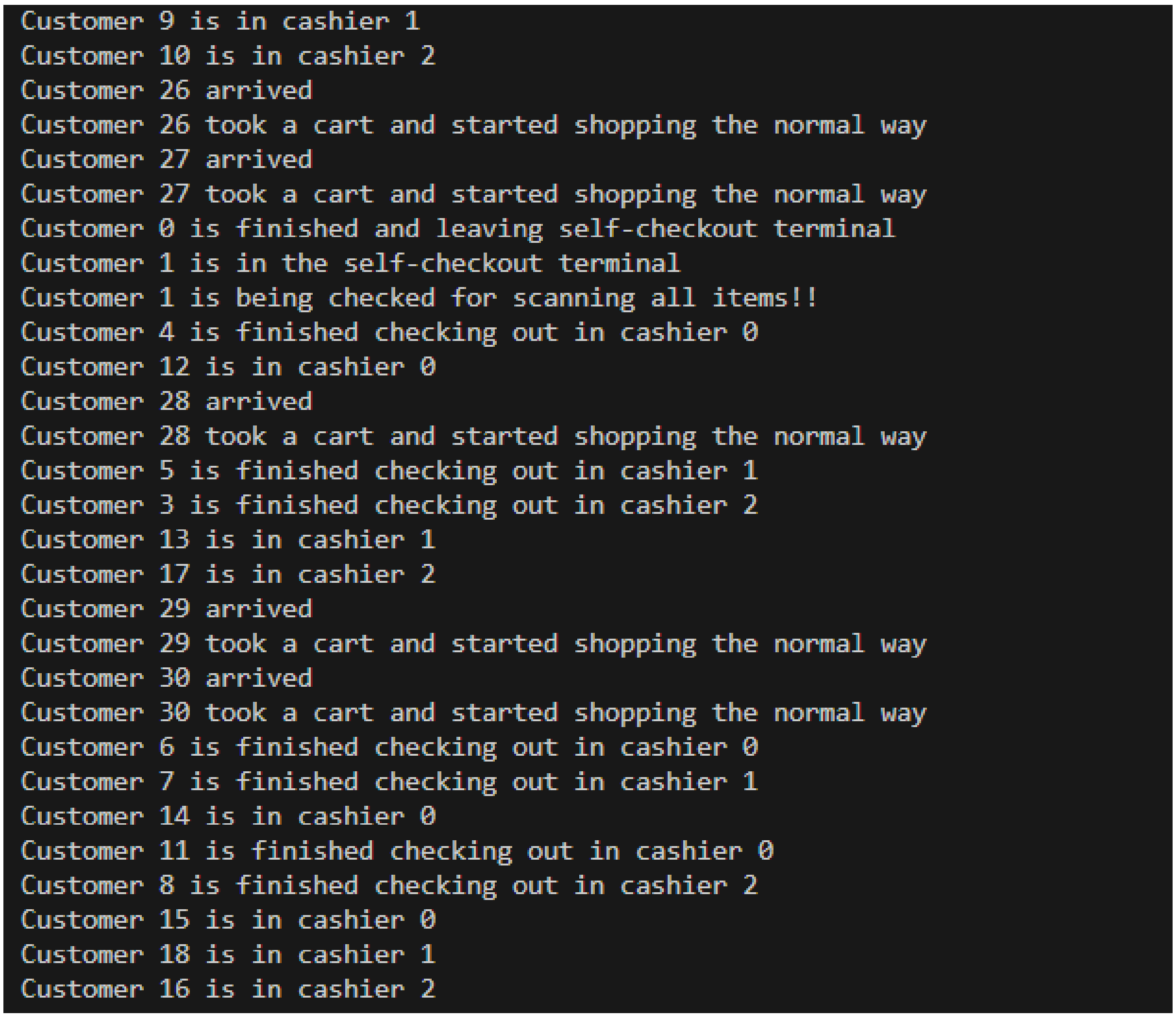


```
abmj01@MSI:/mnt/c/Users/ahmed/second_year/OS2/Download2(1)$ ./sema
Customer 0 arrived
Customer 0 took the handheld scanner
Customer 1 arrived
Customer 1 took the handheld scanner
Customer 2 arrived
Customer 2 took the handheld scanner
Customer 3 arrived
Customer 3 took the handheld scanner
Customer 4 arrived
Customer 4 took the handheld scanner
Customer 5 arrived
Customer 5 took a cart and started shopping the normal way
Moving scanners back to the pick up location!!!!
Customer 6 arrived
Customer 6 took a cart and started shopping the normal way
Customer 7 arrived
Customer 7 took a cart and started shopping the normal way
Customer 8 arrived
Customer 8 took a cart and started shopping the normal way
Customer 9 arrived
Customer 9 took a cart and started shopping the normal way
Customer 10 arrived
Customer 10 took a cart and started shopping the normal way
Customer 11 arrived
Customer 11 took a cart and started shopping the normal way
Customer 12 arrived
Customer 13 arrived
Customer 12 took a cart and started shopping the normal way
Customer 13 took a cart and started shopping the normal way
Customer 14 arrived
Customer 14 took a cart and started shopping the normal way
Customer 15 arrived
Customer 15 took a cart and started shopping the normal way
Customer 16 arrived
Customer 16 took a cart and started shopping the normal way
Customer 17 arrived
```

Figure 3 - customer choose shopping the normal way if there is no scanners.

3- Customers that are doing shopping the normal way switch between cashiers if the queue in a certain cashier is long.

To test this: Reduce the number of scanners and terminals; increase the number of carts:

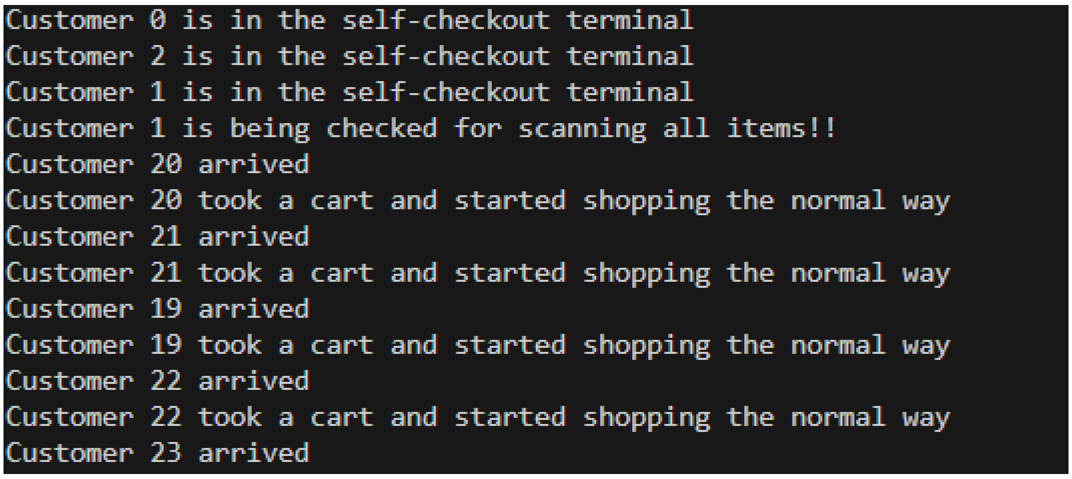


```
Customer 9 is in cashier 1
Customer 10 is in cashier 2
Customer 26 arrived
Customer 26 took a cart and started shopping the normal way
Customer 27 arrived
Customer 27 took a cart and started shopping the normal way
Customer 0 is finished and leaving self-checkout terminal
Customer 1 is in the self-checkout terminal
Customer 1 is being checked for scanning all items!!
Customer 4 is finished checking out in cashier 0
Customer 12 is in cashier 0
Customer 28 arrived
Customer 28 took a cart and started shopping the normal way
Customer 5 is finished checking out in cashier 1
Customer 3 is finished checking out in cashier 2
Customer 13 is in cashier 1
Customer 17 is in cashier 2
Customer 29 arrived
Customer 29 took a cart and started shopping the normal way
Customer 30 arrived
Customer 30 took a cart and started shopping the normal way
Customer 6 is finished checking out in cashier 0
Customer 7 is finished checking out in cashier 1
Customer 14 is in cashier 0
Customer 11 is finished checking out in cashier 0
Customer 8 is finished checking out in cashier 2
Customer 15 is in cashier 0
Customer 18 is in cashier 1
Customer 16 is in cashier 2
```

Figure 4 - customers choosing between different cashiers depending on the queue in each cashier.

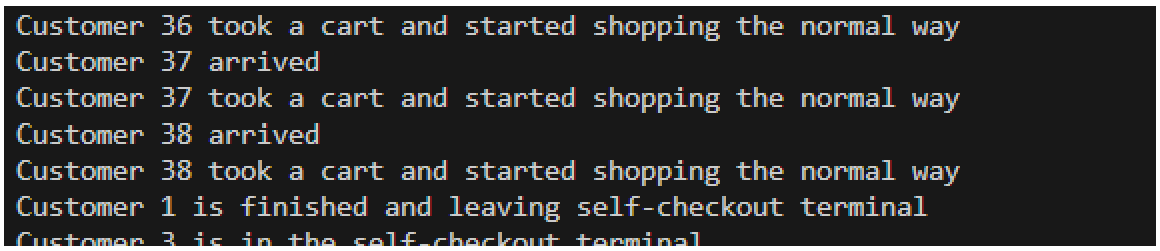
4- Customer checking out in the terminal get control checked randomly:

To test this: Increase the number of scanners to see if there is customer that is going to get control check:



```
Customer 0 is in the self-checkout terminal
Customer 2 is in the self-checkout terminal
Customer 1 is in the self-checkout terminal
Customer 1 is being checked for scanning all items!!
Customer 20 arrived
Customer 20 took a cart and started shopping the normal way
Customer 21 arrived
Customer 21 took a cart and started shopping the normal way
Customer 19 arrived
Customer 19 took a cart and started shopping the normal way
Customer 22 arrived
Customer 22 took a cart and started shopping the normal way
Customer 23 arrived
```

Figure 5 - customer 1 get checked normally and took more time to check out.

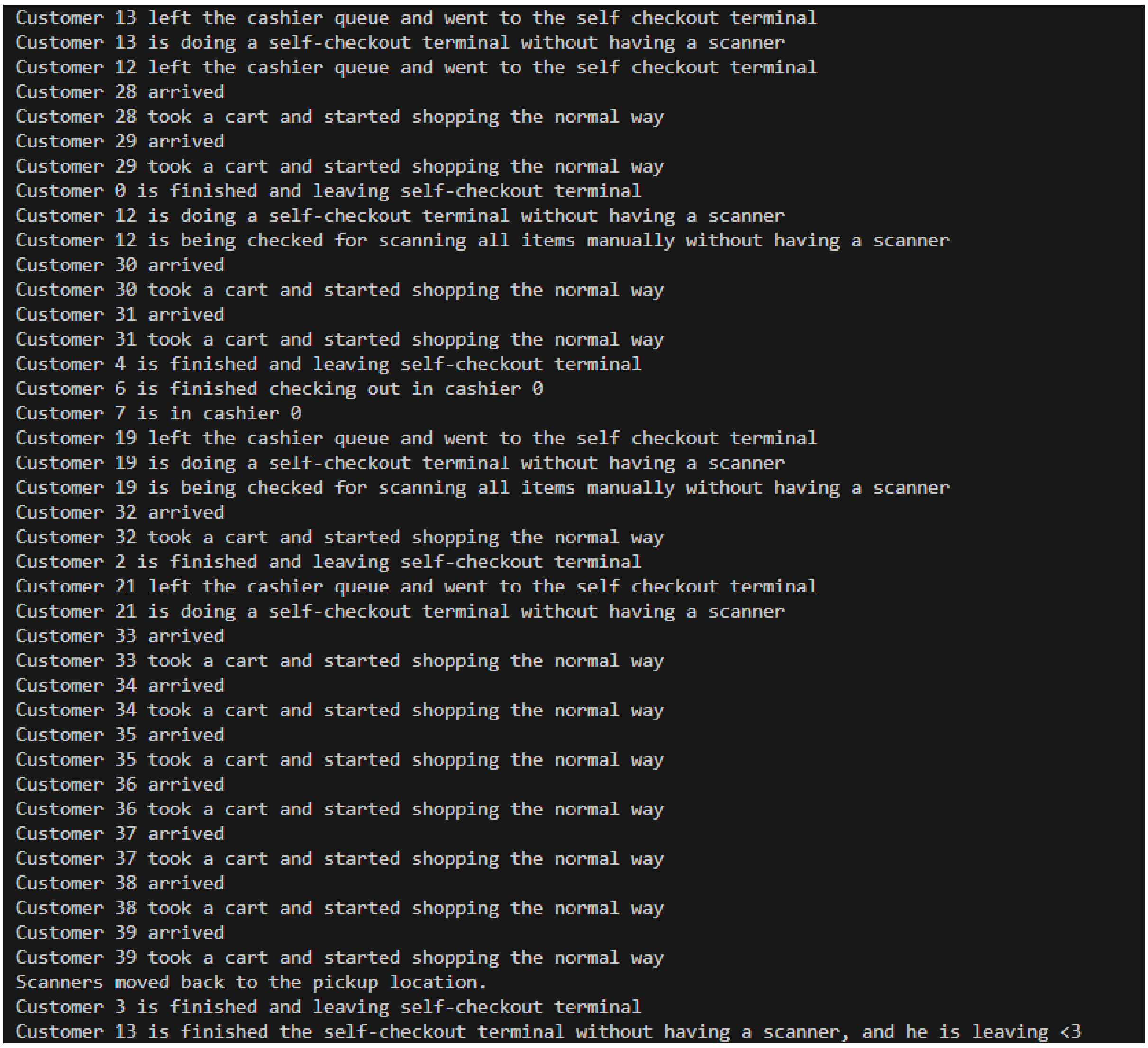


```
Customer 36 took a cart and started shopping the normal way
Customer 37 arrived
Customer 37 took a cart and started shopping the normal way
Customer 38 arrived
Customer 38 took a cart and started shopping the normal way
Customer 1 is finished and leaving self-checkout terminal
Customer 3 is in the self-checkout terminal
```

Figure 6- customer 1 checking out later.

5- Customer without handheld scanner can skip the cashier queue if the queue is long and there is an available self-check terminal.

To test this: Increase the number of terminals and decrease the number of cashiers and scanners:

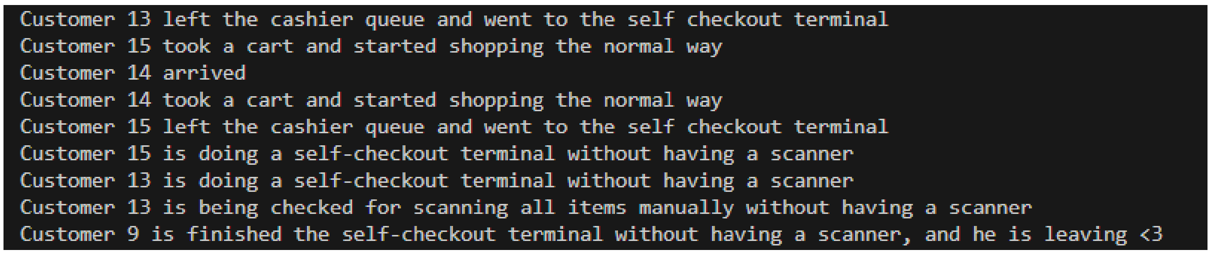


```
Customer 13 left the cashier queue and went to the self checkout terminal
Customer 13 is doing a self-checkout terminal without having a scanner
Customer 12 left the cashier queue and went to the self checkout terminal
Customer 28 arrived
Customer 28 took a cart and started shopping the normal way
Customer 29 arrived
Customer 29 took a cart and started shopping the normal way
Customer 0 is finished and leaving self-checkout terminal
Customer 12 is doing a self-checkout terminal without having a scanner
Customer 12 is being checked for scanning all items manually without having a scanner
Customer 30 arrived
Customer 30 took a cart and started shopping the normal way
Customer 31 arrived
Customer 31 took a cart and started shopping the normal way
Customer 4 is finished and leaving self-checkout terminal
Customer 6 is finished checking out in cashier 0
Customer 7 is in cashier 0
Customer 19 left the cashier queue and went to the self checkout terminal
Customer 19 is doing a self-checkout terminal without having a scanner
Customer 19 is being checked for scanning all items manually without having a scanner
Customer 32 arrived
Customer 32 took a cart and started shopping the normal way
Customer 2 is finished and leaving self-checkout terminal
Customer 21 left the cashier queue and went to the self checkout terminal
Customer 21 is doing a self-checkout terminal without having a scanner
Customer 33 arrived
Customer 33 took a cart and started shopping the normal way
Customer 34 arrived
Customer 34 took a cart and started shopping the normal way
Customer 35 arrived
Customer 35 took a cart and started shopping the normal way
Customer 36 arrived
Customer 36 took a cart and started shopping the normal way
Customer 37 arrived
Customer 37 took a cart and started shopping the normal way
Customer 38 arrived
Customer 38 took a cart and started shopping the normal way
Customer 39 arrived
Customer 39 took a cart and started shopping the normal way
Scanners moved back to the pickup location.
Customer 3 is finished and leaving self-checkout terminal
Customer 13 is finished the self-checkout terminal without having a scanner, and he is leaving <3
```

Figure 7 - customer 13 leaving the cahier queue and going to the self-checkout terminal.

6- Customers who skipped the cashier queue and went to self-checkout terminals also get checked randomly:

To test this: Increase the number of terminals and decrease the number of cashiers, scanners, the time taken to shop in the normal way, and the time taken to control check:

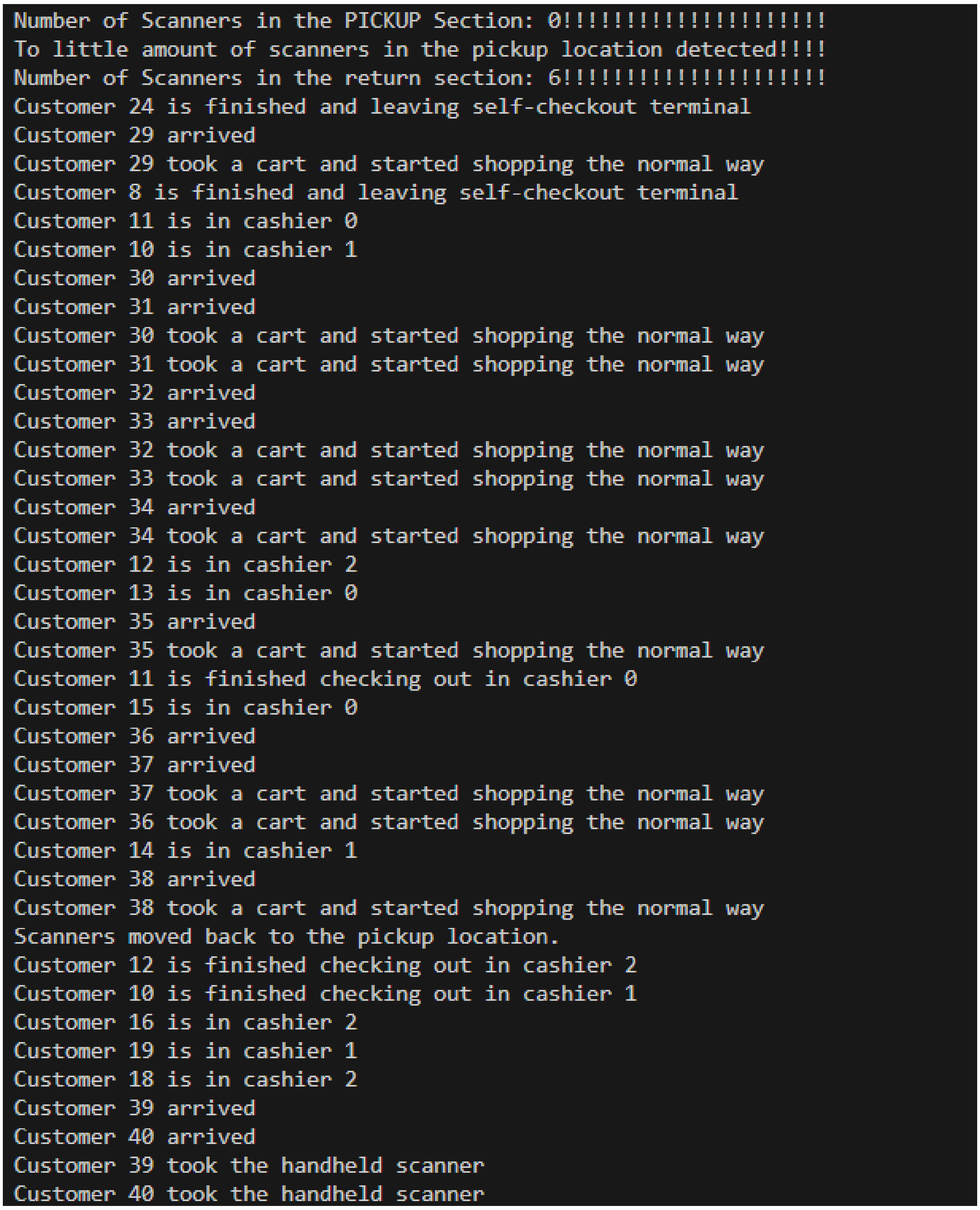


```
Customer 13 left the cashier queue and went to the self checkout terminal
Customer 15 took a cart and started shopping the normal way
Customer 14 arrived
Customer 14 took a cart and started shopping the normal way
Customer 15 left the cashier queue and went to the self checkout terminal
Customer 15 is doing a self-checkout terminal without having a scanner
Customer 13 is doing a self-checkout terminal without having a scanner
Customer 13 is being checked for scanning all items manually without having a scanner
Customer 9 is finished the self-checkout terminal without having a scanner, and he is leaving <3
```

Figure 8 - customer 13 is being control checked while not having a handheld scanner.

7- Handheld scanners return from the return section to the pickup section if the pickup section scanners are below certain threshold and there are enough scanners in the return section:

To test this: Decrease the time taken to shop with a handheld scanner, and decrease the self-checkout time:



```
Number of Scanners in the PICKUP Section: 0!!!!!!!!!!!!!!!!!!!!!
To little amount of scanners in the pickup location detected!!!!
Number of Scanners in the return section: 6!!!!!!!!!!!!!!!!!!!!!
Customer 24 is finished and leaving self-checkout terminal
Customer 29 arrived
Customer 29 took a cart and started shopping the normal way
Customer 8 is finished and leaving self-checkout terminal
Customer 11 is in cashier 0
Customer 10 is in cashier 1
Customer 30 arrived
Customer 31 arrived
Customer 30 took a cart and started shopping the normal way
Customer 31 took a cart and started shopping the normal way
Customer 32 arrived
Customer 33 arrived
Customer 32 took a cart and started shopping the normal way
Customer 33 took a cart and started shopping the normal way
Customer 34 arrived
Customer 34 took a cart and started shopping the normal way
Customer 12 is in cashier 2
Customer 13 is in cashier 0
Customer 35 arrived
Customer 35 took a cart and started shopping the normal way
Customer 11 is finished checking out in cashier 0
Customer 15 is in cashier 0
Customer 36 arrived
Customer 37 arrived
Customer 37 took a cart and started shopping the normal way
Customer 36 took a cart and started shopping the normal way
Customer 14 is in cashier 1
Customer 38 arrived
Customer 38 took a cart and started shopping the normal way
Scanners moved back to the pickup location.
Customer 12 is finished checking out in cashier 2
Customer 10 is finished checking out in cashier 1
Customer 16 is in cashier 2
Customer 19 is in cashier 1
Customer 18 is in cashier 2
Customer 39 arrived
Customer 40 arrived
Customer 39 took the handheld scanner
Customer 40 took the handheld scanner
```

Figure 9 - scanners returned and upcoming customers start getting handheld scanners.

Description of the previous image: As you can see in the beginning of the image the number of scanners in the pickup location went to zero and there was 6 scanners already in the return scanner section, so the scanners was moved successfully and you can see in the end of the image that the upcoming customers took a handheld scanner.

Conclusion:

Every task is accomplished in the assignment including extras, all functionalities work effectively. All of the important tests have been tested and shown above.

Reflection:

This assignment was a lot, but really fun when you understand it and start implementing a testing it. I can safely say that I understand semaphores, thread, and mutexes very well now.