CSE 344

Final Homework Report

Berru Lafcı

1901042681

Compile and clean:

```
berry@DESKTOP-O92GAB6:/mnt/c/Users/lafci/Desktop/GTU/4.
al$ make clean
rm -f PideShop HungryVeryMuch
berry@DESKTOP-O92GAB6:/mnt/c/Users/lafci/Desktop/GTU/4.
al$ make
gcc -o PideShop PideShop.c -pthread -lm
gcc -o HungryVeryMuch HungryVeryMuch.c
berry@DESKTOP-O92GAB6:/mnt/c/Users/lafci/Desktop/GTU/4.
al$
```

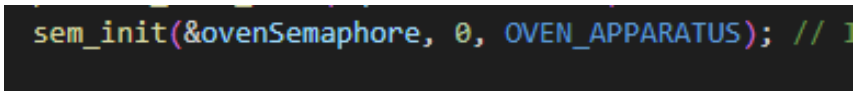
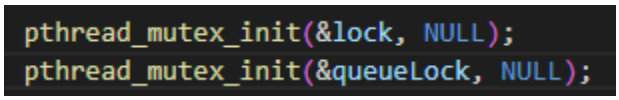
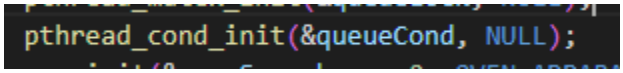
Run:

```
berry@DESKTOP-O92GAB6:/mnt/c/Us
al$ ./PideShop 8080 5 5 5
```

```
./HungryVeryMuch 0.0.0.0 8080 11 50 60
```

Logic of the code:

1. Initialization

- **sem_t ovenSemaphore;**
    - A semaphore is initialized with a value of 3 to represent the 3 oven apparatus (fırıncı küreği) available in the shop. This ensures that only 3 cooking personnel can use the oven at the same time.
    - 
      ```
      sem_init(&ovenSemaphore, 0, OVEN_APPARATUS); // I
      ```
- **pthread_mutex_t lock, queueLock;**
    - Mutexes are used to protect shared resources (like order status and queue size) and ensure mutual exclusion. **lock** is used to protect access to the orders and delivery counts, while **queueLock** is used to manage the order queue.
    - 
      ```
      pthread_mutex_init(&lock, NULL);
      pthread_mutex_init(&queueLock, NULL);
      ```
- **pthread_cond_t queueCond;**
    - A condition variable is used to signal when new orders are added to the queue. This helps cooks wait for new orders efficiently.
    - 
      ```
      pthread_cond_init(&queueCond, NULL);
      ```

- **Structures**

```c
// Structures for cooks, delivery personnel, and orders
typedef struct {
    int id;
    pthread_t thread;
    int capacity;
    int orders[3];
    int order_count;
    int delivery_count; // To find promoted delivery person
} DeliveryPerson;

typedef struct {
    int id;
    pthread_t thread;
} Cook;

typedef struct {
    int order_id;
    int customer_id;
    int status; // 0: placed, 1: prepared, 2: cooked, 3: delivering
    int x;
    int y;
    int distance;
} Order;
```

- **Pool threads:**

```c
for (int i = 0; i < cookPoolSize; i++) {
    cooks[i].id = i;
    pthread_create(&cooks[i].thread, NULL, cook_function, (void *)&cooks[i]);
}

for (int i = 0; i < deliveryPoolSize; i++) {
    delivery[i].id = i;
    delivery[i].capacity = 3;
    delivery[i].order_count = 0;
    delivery[i].delivery_count = 0;
    pthread_create(&delivery[i].thread, NULL, delivery_function, (void *)&delivery[i]);
}
```

2. Order Handling

Adding Orders to the Queue:

```
void add_order(Order order) {
    pthread_mutex_lock(&queueLock);
    if (queueSize < QUEUE_SIZE) {
        orderQueue[queueSize++] = order;
        pthread_cond_signal(&queueCond);
    }
    pthread_mutex_unlock(&queueLock);
}
```

- Locks the queue and enters the critical region. Ensures the queue isn't full before adding a new order. Adds the new order to the queue and increments the queue size. Signals any waiting cooks that a new order is available. Unlocks the queue, allowing other threads to access it.

Getting Orders from the Queue:

```
Order get_order() {
    pthread_mutex_lock(&queueLock);
    while (queueSize == 0) {
        pthread_cond_wait(&queueCond, &queueLock);
    }
    Order order = orderQueue[--queueSize];
    pthread_mutex_unlock(&queueLock);
    return order;
}
```

- This function locks the queueLock mutex and waits to use pthread_cond_wait if the queue is empty. When the order adds to the queue and the queueSize increase, pthread_cond_signal signals it and exit from the loop. Once an order is available, it is removed from the queue, and the queueLock mutex is unlocked.

3. Cook Synchronization

Cook Function:

-
```
2
3   void *cook_function(void *arg) {
4       Cook *cook = (Cook *)arg;
5       while (1) {
6           Order order = get_order();
7
```
- Each cook thread waits for an order from the queue using get_order.
- After preparing the order (simulated by pseudo-inverse time), the cook waits for an oven apparatus to become available using sem_wait(&ovenSemaphore). There are 3 apparatus at most.
-
```
double preparation_time = calculate_pseudo_inverse();
sleep((int)preparation_time); // Simulate preparation time
```
- The order is then cooked (simulated by half of the pseudo-inverse time), and the cook updates the order status and places it back in the global orders array.
-
```
sleep((int)(preparation_time / 2)); // Sim
```
- Then it changes its status to 2 as cooked. And because of MAX_OVEN is 6 (which shows the oven can hold 6 meals inside) it takes % with it.
-
```
pthread_mutex_lock(&lock);
order.status = 2; // Cooked
orders[order.order_id % MAX_OVEN] = order; // Assign the order back to the global orders array
pthread_mutex_unlock(&lock);
```
- The oven apparatus is released using sem_post(&ovenSemaphore).

- **Calculating pseudo-inverse:**
  - It initializes a matrix A with random complex numbers. It calculates the conjugate transpose of matrix A and stores it in A_conj_trans. It performs matrix multiplication and inversion to calculate the pseudo-inverse of A and stores it in A_pseudo_inv.
  - It simulates computation time by pausing execution for 1 second using the sleep() function. If I don't add sleep(1) it passes so fast.

- It measures the elapsed time using the gettimeofday() function then it returns the elapsed time as a double value.

```
sleep(1); // Simulate computation time

gettimeofday(&end, NULL);
double elapsed_time = (end.tv_sec - start.tv_sec) * 1.0 + (end.tv_usec - start.tv_usec) / 1000000.0;

return elapsed_time;
```

4. Delivery Synchronization

Delivery Function:

```
void *delivery_function(void *arg) {

    DeliveryPerson *delivery = (DeliveryPerson *)arg; // Get the delivery person from the argument

    while (1) {
        pthread_mutex_lock(&lock);
```

- It locks the lock mutex to safely access and update the global orders array.
- Each delivery thread checks for cooked orders that are ready to be delivered with checking **order[i].status** as 2.
- The delivery persons capacity is 3 orders. But to also handle the last 1 or 2 orders, it checks the **order_count < capacity**. After the checks, it updates the status as 3 to make it delivering.

```
for (int i = 0; i < MAX_OVEN; i++) {
    if (orders[i].status == 2 && delivery->order_count < delivery->capacity) {
```

- If no cooked orders are found (found_order == 0), the delivery person sleeps for 1 second and then continues to the next iteration to check for orders again.

```
if (!found_order) {
    sleep(1); // Wait for orders
    continue;
}
```

- For each order, it calculates the Euclidean distance to the customer's location using the coordinates order.x and order.y. The delivery time is simulated by sleeping for distance / deliverySpeed (from argument) seconds.

- Finally, the **notify_client** function sends a message back to the client to notify them that their order has been delivered. It increments the delivered_orders counter and checks if all orders have been delivered. If all orders are delivered, it closes the client socket and resets the delivered_orders counter for the next client.

```c
void notify_client(int order_id) {
    pthread_mutex_lock(&lock);
    delivered_orders++;
    pthread_mutex_unlock(&lock);

    char message[256];
    snprintf(message, sizeof(message), "Order %d has been delivered.\n", order_id);
    send(client_socket, message, strlen(message), 0);

    if (delivered_orders == total_orders) {
        close(client_socket);
        delivered_orders = 0; // Reset for the next client
    }
}
```

6. Handling Multiple Clients

Client Handler:

```c
void* client_handler(void* arg) {
    int new_socket = *((int*)arg);
    int order_count = 0;
```

- This function is executed in a separate thread for each client. It receives client pid and orders from the client, adds them to the queue, and then closes the client connection.

Main Function:

- The main function sets up the server and continuously waits for new client connections. For each new connection, a new client handler thread is created to process the client's orders.

7. Signal Handling and Cleanup

```c
void handle_signal(int signal) {
    if (signal == SIGINT || signal == SIGTERM) {
        printf("Quiting...writing log file\n");
        log_activity("Server shutting down...");
        promote_delivery_person();

        // Cleanup
        pthread_mutex_destroy(&lock);
        pthread_mutex_destroy(&queueLock);
        sem_destroy(&ovenSemaphore); // Destroy semaphore
        close(client_socket);

        // Clean up threads
        for (int i = 0; i < cookPoolSize; i++) {
            pthread_cancel(cooks[i].thread);
        }

        for (int i = 0; i < deliveryPoolSize; i++) {
            pthread_cancel(delivery[i].thread);
        }

        // Exit the program
        exit(0);
    }
}
```

At the end of the day, when we close the server, we can see the promoted delivery person. İt calculated with the delivery_count increasing in the delivery_function.

```c
for (int i = 0; i < deliveryPoolSize; i++) {
    if (delivery[i].delivery_count > max_deliveries) {
        max_deliveries = delivery[i].delivery_count;
        best_delivery_person = i;
    }
}
```

Socket Communication

The socket function creates a new socket using the AF_INET address family (IPv4) and SOCK_STREAM type (TCP).

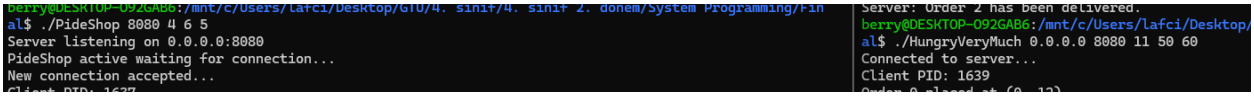
```
if ((server_fd = socket(AF_INET, SOCK_STREAM, 0)) == 0) {
    perror("socket failed");
    exit(EXIT_FAILURE);
}
```

The bind function assigns a local address to the socket. The address structure specifies the IP address and the port number. I generally gave 8080 as localhost port number as argument.

```
if (bind(server_fd, (struct sockaddr *)&address, sizeof(address)) < 0) {
    perror("bind failed");
    exit(EXIT_FAILURE);
}

if (listen(server_fd, 3) < 0) {
    perror("listen");
    exit(EXIT_FAILURE);
}
```

```
serv_addr.sin_family = AF_INET;
serv_addr.sin_port = htons(port);

if (inet_pton(AF_INET, ipAddress, &serv_addr.sin_addr) <= 0) {
    printf("\nInvalid address/ Address not supported \n");
    return -1;
}

if (connect(sock, (struct sockaddr *)&serv_addr, sizeof(serv_addr)) < 0) {
    printf("\nConnection Failed \n");
    return -1;
}

printf("Connected to server...\n");
log_activity("Connected to server...\n");
```

After the server started, I give the corresponding ip address as argument:



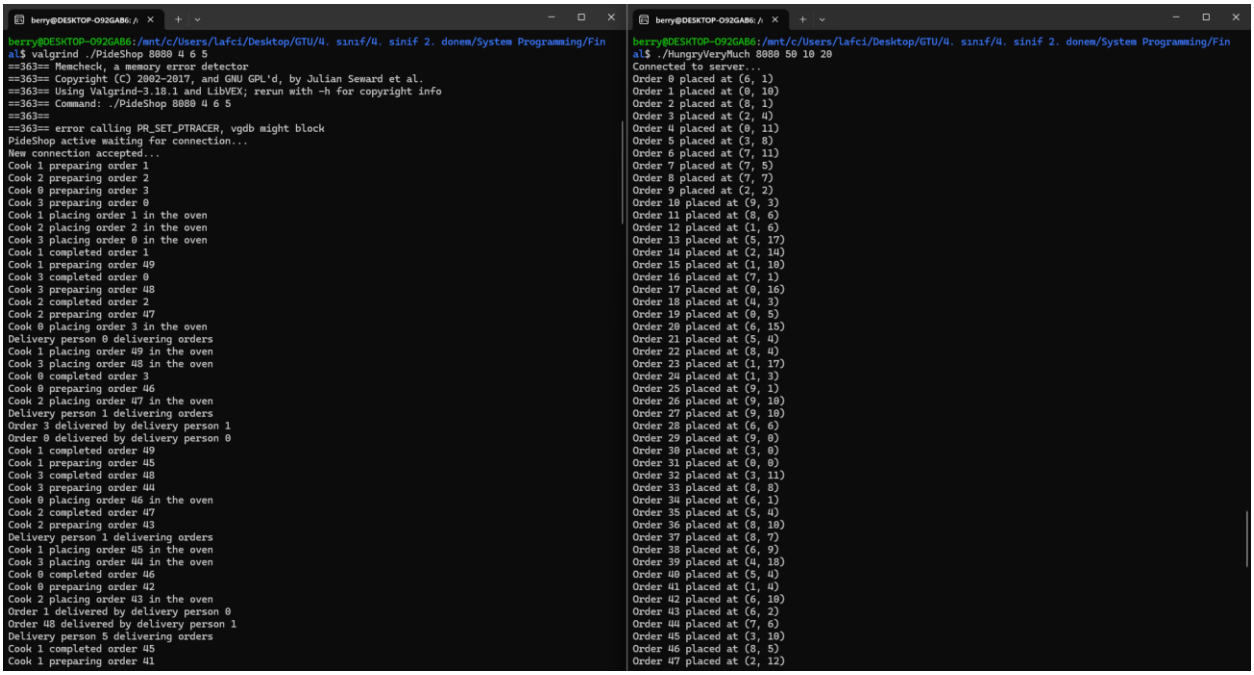
Potential Deadlocks:

Sometimes my code freezes in large order numbers. There may be a deadlock when that happens. Here are the potential deadlock scenarios.

- If **Cook 1** tries to place the order in the oven (holding queueLock and waiting for lock) and **Cook 2** tries to access the oven (waiting for queueLock), both cooks are stuck, leading to a deadlock.
- If **Cook 1** is waiting for lock to place a cooked order in the oven and **Delivery Person 1** is trying to update the delivery status (holding lock), both are stuck, leading to a deadlock.

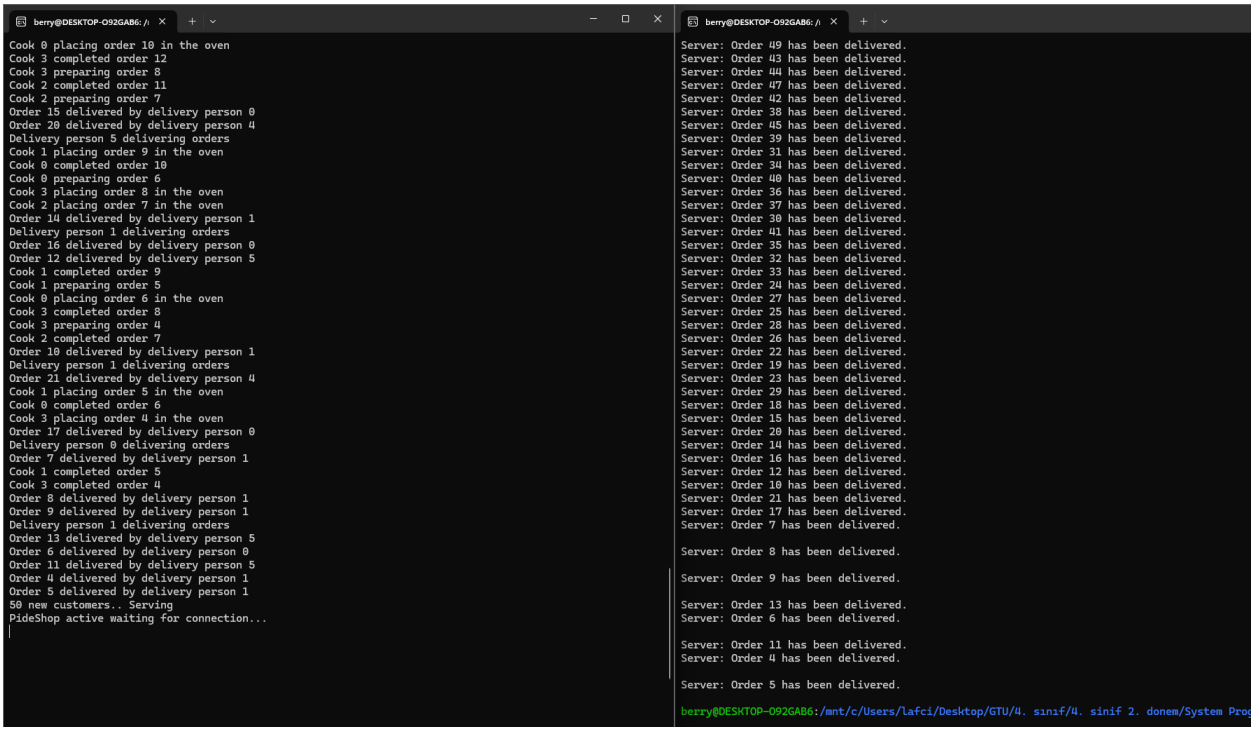
Test Cases:

1-) 50 orders:



```
berry@DESKTOP-O92GAB6:/mnt/c/Users/lafci/Desktop/GTU/4. sınıf/4. sinif 2. donem/System Programming/Fin
al$ valgrind ./PideShop 8080 4 6 5
==363== Memcheck, a memory error detector
==363== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==363== Using Valgrind-3.18.1 and LibVEX; rerun with -h for copyright info
==363== Command: ./PideShop 8080 4 6 5
==363==
==363== error calling PR_SET_PTRACER, vgdb might block
PideShop active waiting for connection...
New connection accepted...
Cook 1 preparing order 1
Cook 2 preparing order 2
Cook 0 preparing order 3
Cook 3 preparing order 0
Cook 1 placing order 1 in the oven
Cook 2 placing order 2 in the oven
Cook 3 placing order 0 in the oven
Cook 1 completed order 1
Cook 1 preparing order 49
Cook 3 completed order 0
Cook 3 preparing order 48
Cook 2 completed order 2
Cook 2 preparing order 47
Cook 0 placing order 3 in the oven
Delivery person 0 delivering orders
Cook 1 placing order 49 in the oven
Cook 3 placing order 48 in the oven
Cook 0 completed order 3
Cook 0 preparing order 46
Cook 2 placing order 47 in the oven
Delivery person 1 delivering orders
Order 3 delivered by delivery person 1
Order 0 delivered by delivery person 0
Cook 1 completed order 49
Cook 1 preparing order 45
Cook 3 completed order 48
Cook 3 preparing order 44
Cook 0 placing order 46 in the oven
Cook 2 completed order 47
Cook 2 preparing order 43
Delivery person 1 delivering orders
Cook 1 placing order 45 in the oven
Cook 3 placing order 44 in the oven
Cook 0 completed order 46
Cook 0 preparing order 42
Cook 2 placing order 43 in the oven
Order 1 delivered by delivery person 0
Order 48 delivered by delivery person 1
Delivery person 5 delivering orders
Cook 1 completed order 45
Cook 1 preparing order 41
```

```
berry@DESKTOP-O92GAB6:/mnt/c/Users/lafci/Desktop/GTU/4. sınıf/4. sinif 2. donem/System Programming/Fin
al$ ./HungryVeryMuch 8080 50 10 20
Connected to server...
Order 0 placed at (6, 1)
Order 1 placed at (0, 10)
Order 2 placed at (8, 1)
Order 3 placed at (2, 4)
Order 4 placed at (0, 11)
Order 5 placed at (3, 8)
Order 6 placed at (7, 11)
Order 7 placed at (7, 5)
Order 8 placed at (7, 7)
Order 9 placed at (2, 2)
Order 10 placed at (9, 3)
Order 11 placed at (8, 6)
Order 12 placed at (1, 6)
Order 13 placed at (5, 17)
Order 14 placed at (2, 14)
Order 15 placed at (1, 10)
Order 16 placed at (7, 1)
Order 17 placed at (0, 16)
Order 18 placed at (4, 3)
Order 19 placed at (0, 5)
Order 20 placed at (6, 15)
Order 21 placed at (5, 4)
Order 22 placed at (8, 4)
Order 23 placed at (1, 17)
Order 24 placed at (1, 3)
Order 25 placed at (9, 1)
Order 26 placed at (9, 10)
Order 27 placed at (9, 10)
Order 28 placed at (6, 6)
Order 29 placed at (9, 0)
Order 30 placed at (3, 0)
Order 31 placed at (0, 0)
Order 32 placed at (3, 11)
Order 33 placed at (8, 8)
Order 34 placed at (6, 1)
Order 35 placed at (5, 4)
Order 36 placed at (8, 10)
Order 37 placed at (8, 7)
Order 38 placed at (6, 9)
Order 39 placed at (4, 18)
Order 40 placed at (5, 4)
Order 41 placed at (1, 4)
Order 42 placed at (6, 10)
Order 43 placed at (6, 2)
Order 44 placed at (7, 6)
Order 45 placed at (3, 10)
Order 46 placed at (8, 5)
Order 47 placed at (2, 12)
```



```
Cook 0 placing order 10 in the oven
Cook 3 completed order 12
Cook 3 preparing order 8
Cook 2 completed order 11
Cook 2 preparing order 7
Order 15 delivered by delivery person 0
Order 20 delivered by delivery person 4
Delivery person 5 delivering orders
Cook 1 placing order 9 in the oven
Cook 0 completed order 10
Cook 0 preparing order 6
Cook 3 placing order 8 in the oven
Cook 2 placing order 7 in the oven
Order 14 delivered by delivery person 1
Delivery person 1 delivering orders
Order 16 delivered by delivery person 0
Order 12 delivered by delivery person 5
Cook 1 completed order 9
Cook 1 preparing order 5
Cook 0 placing order 6 in the oven
Cook 3 completed order 8
Cook 3 preparing order 4
Cook 2 completed order 7
Order 10 delivered by delivery person 1
Delivery person 1 delivering orders
Order 21 delivered by delivery person 4
Cook 1 placing order 5 in the oven
Cook 0 completed order 6
Cook 3 placing order 4 in the oven
Order 17 delivered by delivery person 0
Delivery person 0 delivering orders
Order 7 delivered by delivery person 1
Cook 1 completed order 5
Cook 3 completed order 4
Order 8 delivered by delivery person 1
Order 9 delivered by delivery person 1
Delivery person 1 delivering orders
Order 13 delivered by delivery person 5
Order 6 delivered by delivery person 0
Order 11 delivered by delivery person 5
Order 4 delivered by delivery person 1
Order 5 delivered by delivery person 1
50 new customers.. Serving
PideShop active waiting for connection...
```

```
Server: Order 49 has been delivered.
Server: Order 43 has been delivered.
Server: Order 44 has been delivered.
Server: Order 47 has been delivered.
Server: Order 42 has been delivered.
Server: Order 38 has been delivered.
Server: Order 45 has been delivered.
Server: Order 39 has been delivered.
Server: Order 31 has been delivered.
Server: Order 34 has been delivered.
Server: Order 40 has been delivered.
Server: Order 36 has been delivered.
Server: Order 37 has been delivered.
Server: Order 30 has been delivered.
Server: Order 41 has been delivered.
Server: Order 35 has been delivered.
Server: Order 32 has been delivered.
Server: Order 33 has been delivered.
Server: Order 24 has been delivered.
Server: Order 27 has been delivered.
Server: Order 25 has been delivered.
Server: Order 28 has been delivered.
Server: Order 26 has been delivered.
Server: Order 22 has been delivered.
Server: Order 19 has been delivered.
Server: Order 23 has been delivered.
Server: Order 29 has been delivered.
Server: Order 18 has been delivered.
Server: Order 15 has been delivered.
Server: Order 20 has been delivered.
Server: Order 14 has been delivered.
Server: Order 16 has been delivered.
Server: Order 12 has been delivered.
Server: Order 10 has been delivered.
Server: Order 21 has been delivered.
Server: Order 17 has been delivered.
Server: Order 7 has been delivered.

Server: Order 8 has been delivered.

Server: Order 9 has been delivered.

Server: Order 13 has been delivered.
Server: Order 6 has been delivered.

Server: Order 11 has been delivered.
Server: Order 4 has been delivered.

Server: Order 5 has been delivered.

berry@DESKTOP-O92GAB6:/mnt/c/Users/lafci/Desktop/GTU/4. sınıf/4. sinif 2. donem/System Prog
```

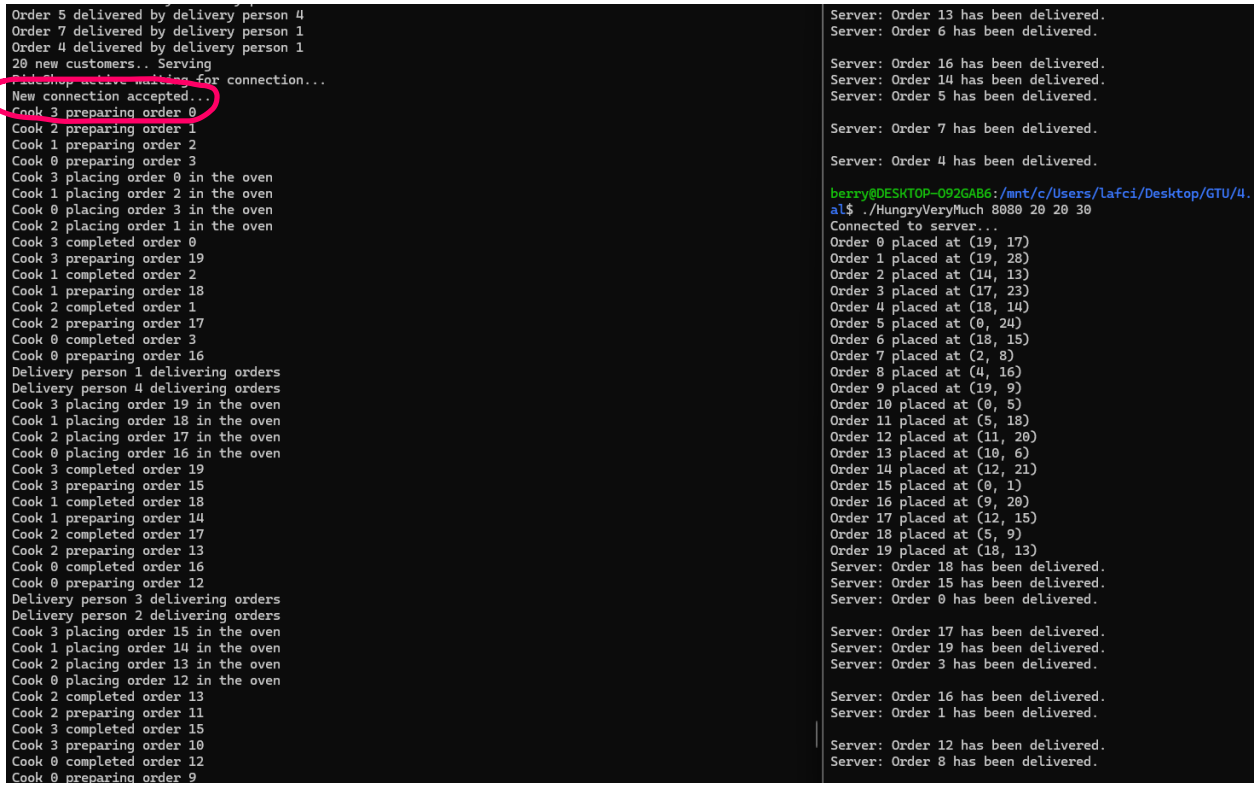
```
^CServer shutting down...
Cleaning up...
Lock destroyed...
Queue lock destroyed...
Queue condition destroyed...
Oven semaphore destroyed...
Destroying threads...
Cooks destroyed...
Delivery personnel destroyed...
==363==
==363== HEAP SUMMARY:
==363==     in use at exit: 3,264 bytes in 12 blocks
==363==   total heap usage: 22 allocs, 10 frees, 16,030 bytes allocated
==363==
==363== LEAK SUMMARY:
==363==    definitely lost: 0 bytes in 0 blocks
==363==    indirectly lost: 0 bytes in 0 blocks
==363==      possibly lost: 3,264 bytes in 12 blocks
==363==    still reachable: 0 bytes in 0 blocks
==363==         suppressed: 0 bytes in 0 blocks
==363== Rerun with --leak-check=full to see details of leaked memory
==363==
==363== For lists of detected and suppressed errors, rerun with: -s
==363== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
berry@DESKTOP-O92GAB6:/mnt/c/Users/lafci/Desktop/GTU/4. sınıf/4. sinif 2. donem/Sys
al$
```

2-) Waiting for new client:

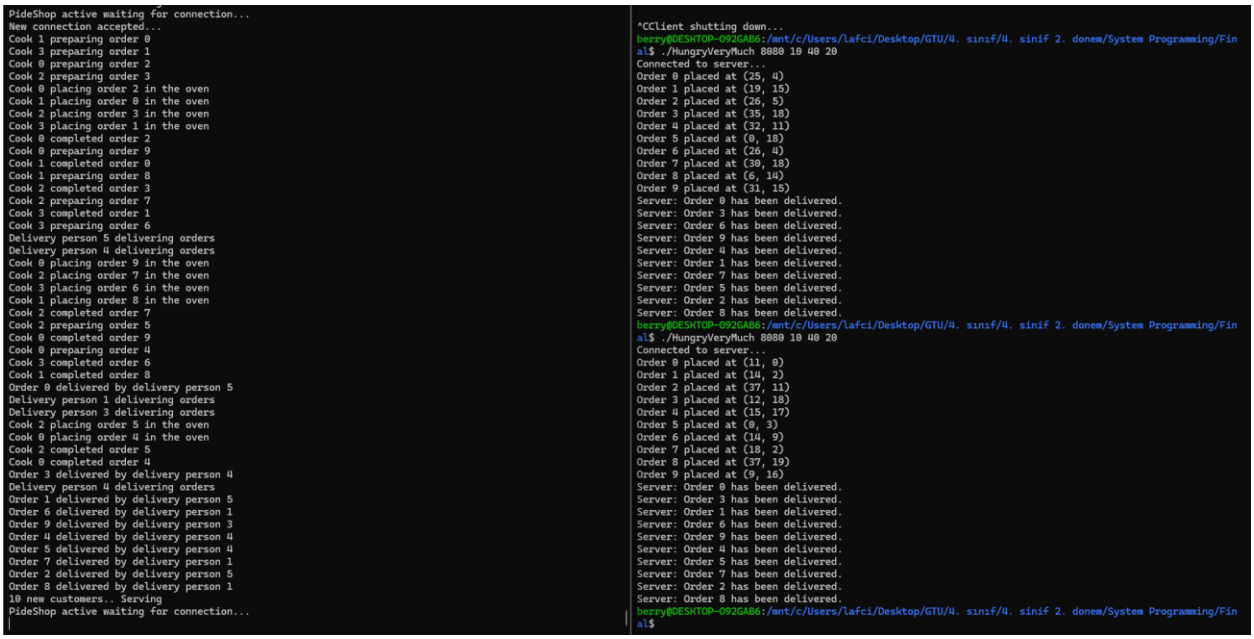
```
Order 11 delivered by delivery person 0
Order 13 delivered by delivery person 5
Order 2 delivered by delivery person 1
Order 6 delivered by delivery person 4
Order 5 delivered by delivery person 3
Order 7 delivered by delivery person 4
Order 9 delivered by delivery person 2
Order 14 delivered by delivery person 5
Order 4 delivered by delivery person 4
Order 10 delivered by delivery person 2
20 new customers.. Serving
PideShop active waiting for connection...
```

```
Server: Order 5 has been delivered.

Server: Order 7 has been delivered.

Server: Order 9 has been delivered.

Server: Order 14 has been delivered.
Server: Order 4 has been delivered.

Server: Order 10 has been delivered.
berry@DESKTOP-O92GAB6:/mnt/c/Users/lafci/Deskto
al$
```

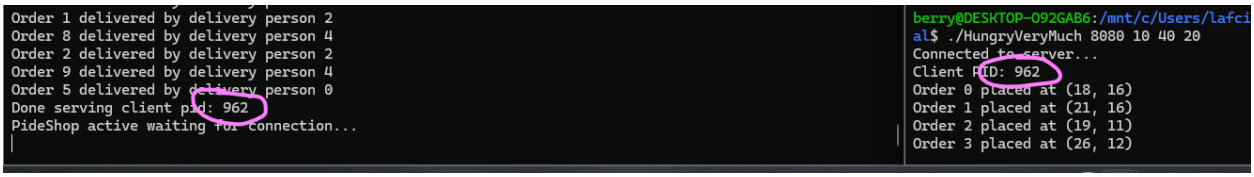
3-) Adding new client:


Order 5 delivered by delivery person 4
Order 7 delivered by delivery person 1
Order 4 delivered by delivery person 1
20 new customers.. Serving
New connection accepted...
Cook 3 preparing order 0
Cook 2 preparing order 1
Cook 1 preparing order 2
Cook 0 preparing order 3
Cook 3 placing order 0 in the oven
Cook 1 placing order 2 in the oven
Cook 0 placing order 3 in the oven
Cook 2 placing order 1 in the oven
Cook 3 completed order 0
Cook 3 preparing order 19
Cook 1 completed order 2
Cook 1 preparing order 18
Cook 2 completed order 1
Cook 2 preparing order 17
Cook 0 completed order 3
Cook 0 preparing order 16
Delivery person 1 delivering orders
Delivery person 4 delivering orders
Cook 3 placing order 19 in the oven
Cook 1 placing order 18 in the oven
Cook 2 placing order 17 in the oven
Cook 0 placing order 16 in the oven
Cook 3 completed order 19
Cook 3 preparing order 15
Cook 1 completed order 18
Cook 1 preparing order 14
Cook 2 completed order 17
Cook 2 preparing order 13
Cook 0 completed order 16
Cook 0 preparing order 12
Delivery person 3 delivering orders
Delivery person 2 delivering orders
Cook 3 placing order 15 in the oven
Cook 1 placing order 14 in the oven
Cook 2 placing order 13 in the oven
Cook 0 placing order 12 in the oven
Cook 2 completed order 13
Cook 2 preparing order 11
Cook 3 completed order 15
Cook 3 preparing order 10
Cook 0 completed order 12
Cook 0 preparing order 9
Server: Order 13 has been delivered.
Server: Order 6 has been delivered.
Server: Order 16 has been delivered.
Server: Order 14 has been delivered.
Server: Order 5 has been delivered.
Server: Order 7 has been delivered.
Server: Order 4 has been delivered.
berry@DESKTOP-O92GAB6:/mnt/c/Users/lafci/Desktop/GTU/4.
al$ ./HungryVeryMuch 8080 20 20 30
Connected to server...
Order 0 placed at (19, 17)
Order 1 placed at (19, 28)
Order 2 placed at (14, 13)
Order 3 placed at (17, 23)
Order 4 placed at (18, 14)
Order 5 placed at (0, 24)
Order 6 placed at (18, 15)
Order 7 placed at (2, 8)
Order 8 placed at (4, 16)
Order 9 placed at (19, 9)
Order 10 placed at (0, 5)
Order 11 placed at (5, 18)
Order 12 placed at (11, 20)
Order 13 placed at (10, 6)
Order 14 placed at (12, 21)
Order 15 placed at (0, 1)
Order 16 placed at (9, 20)
Order 17 placed at (12, 15)
Order 18 placed at (5, 9)
Order 19 placed at (18, 13)
Server: Order 18 has been delivered.
Server: Order 15 has been delivered.
Server: Order 0 has been delivered.
Server: Order 17 has been delivered.
Server: Order 19 has been delivered.
Server: Order 3 has been delivered.
Server: Order 16 has been delivered.
Server: Order 1 has been delivered.
Server: Order 12 has been delivered.
Server: Order 8 has been delivered.

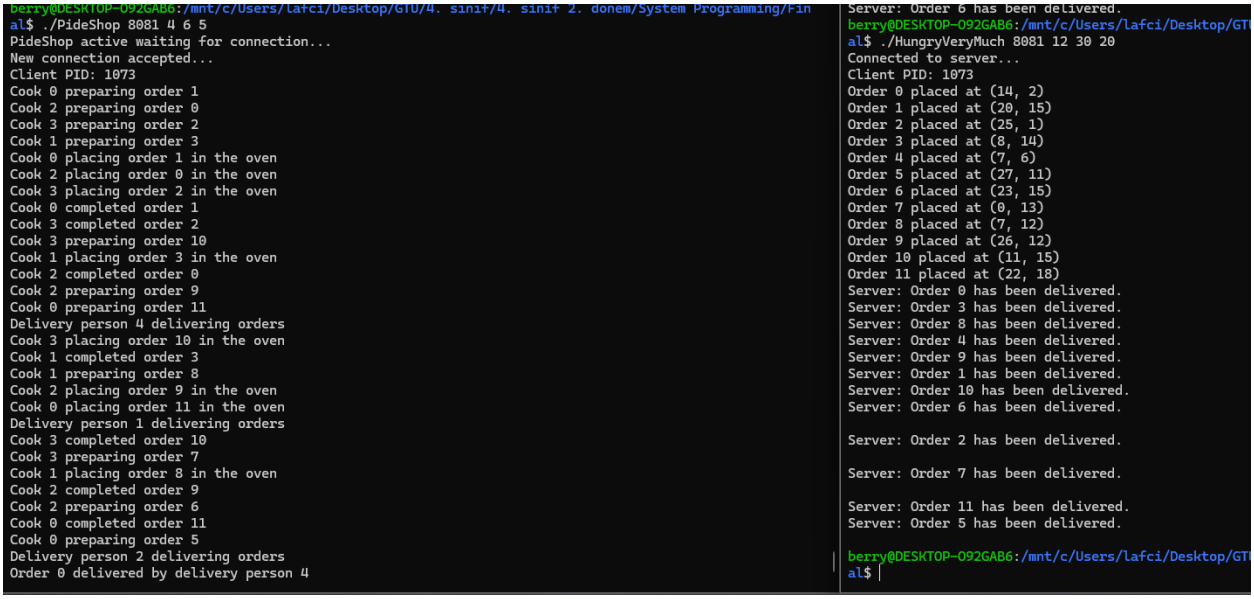
4-) Running 2 client

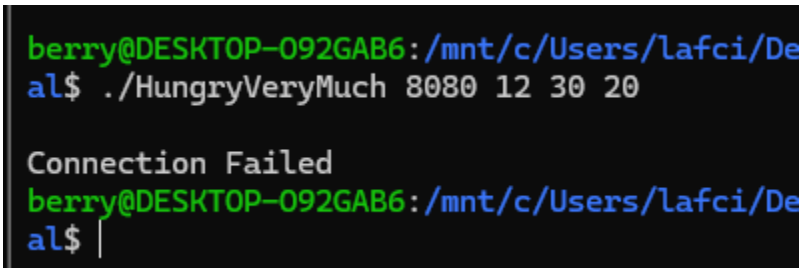


5-) Sending client pid to the server:



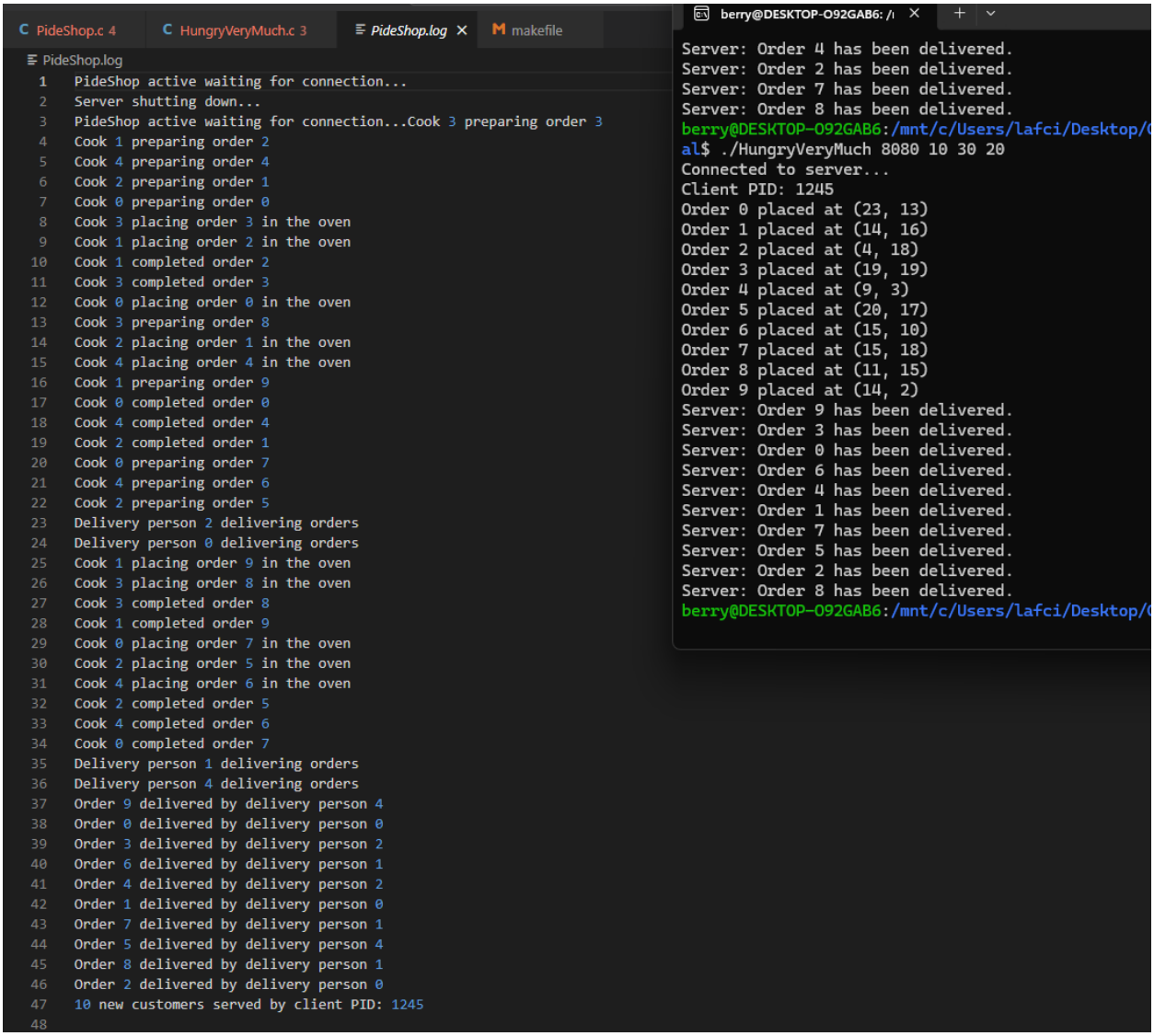
6-) Different port number than 8080:





7-) Trying with different pool sizes:

8-) Log file outputs:



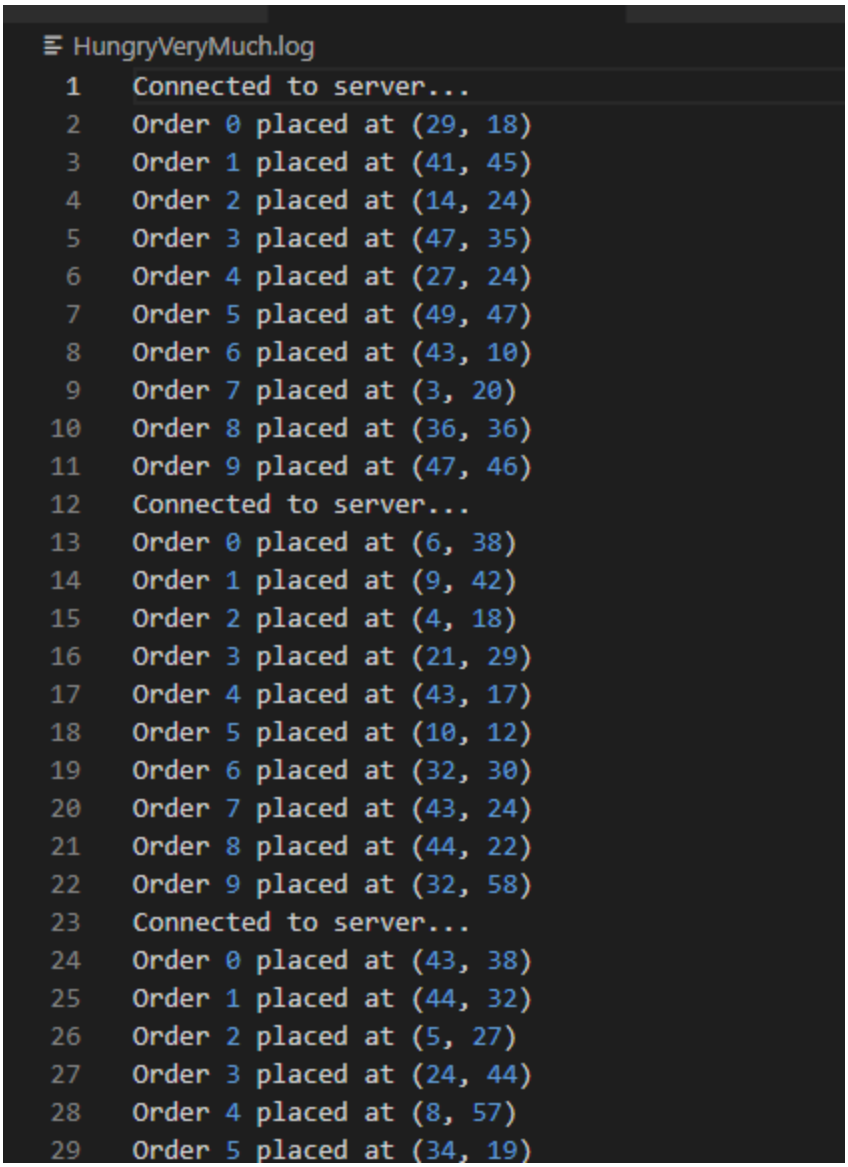
PideShop.c 4 | HungryVeryMuch.c 3 | PideShop.log | makefile

PideShop.log

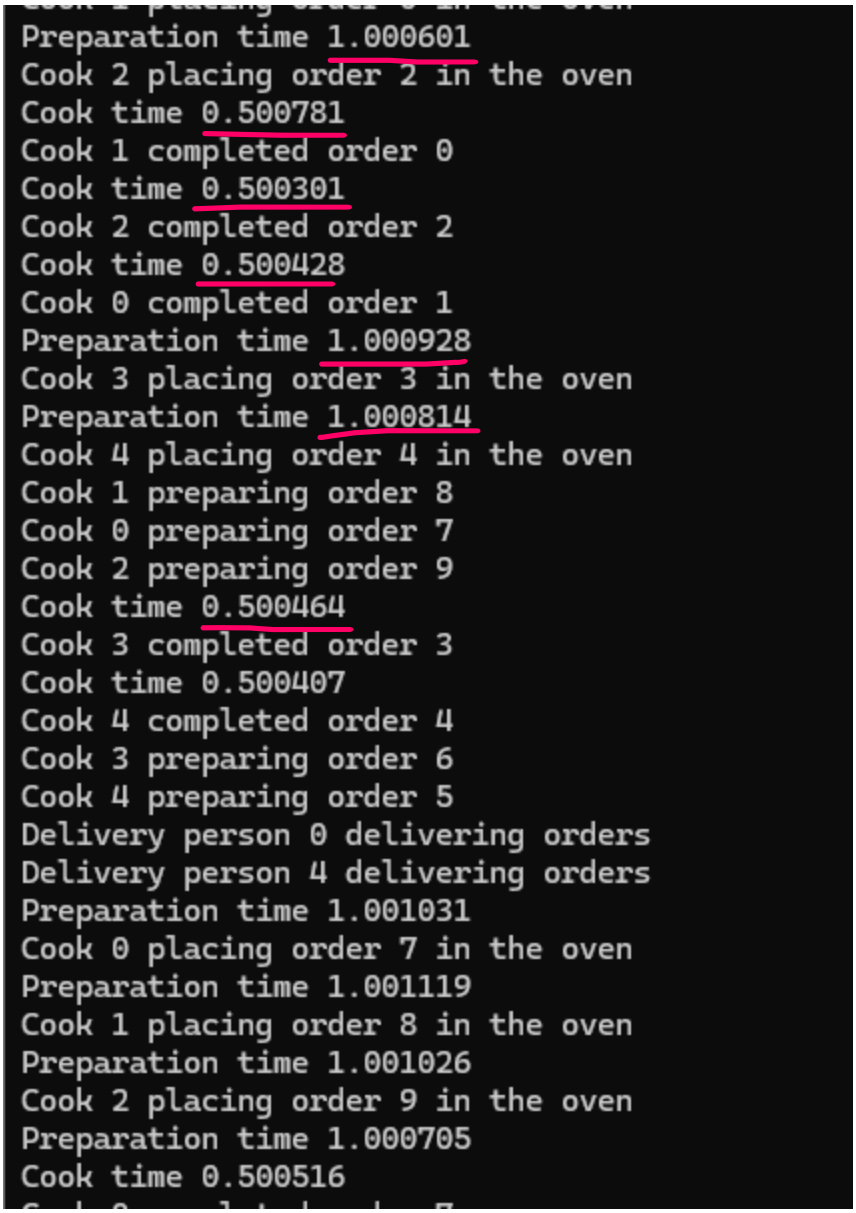
```
PideShop active waiting for connection...
Server shutting down...
PideShop active waiting for connection...Cook 3 preparing order 3
Cook 1 preparing order 2
Cook 4 preparing order 4
Cook 2 preparing order 1
Cook 0 preparing order 0
Cook 3 placing order 3 in the oven
Cook 1 placing order 2 in the oven
Cook 1 completed order 2
Cook 3 completed order 3
Cook 0 placing order 0 in the oven
Cook 3 preparing order 8
Cook 2 placing order 1 in the oven
Cook 4 placing order 4 in the oven
Cook 1 preparing order 9
Cook 0 completed order 0
Cook 4 completed order 4
Cook 2 completed order 1
Cook 0 preparing order 7
Cook 4 preparing order 6
Cook 2 preparing order 5
Delivery person 2 delivering orders
Delivery person 0 delivering orders
Cook 1 placing order 9 in the oven
Cook 3 placing order 8 in the oven
Cook 3 completed order 8
Cook 1 completed order 9
Cook 0 placing order 7 in the oven
Cook 2 placing order 5 in the oven
Cook 4 placing order 6 in the oven
Cook 2 completed order 5
Cook 4 completed order 6
Cook 0 completed order 7
Delivery person 1 delivering orders
Delivery person 4 delivering orders
Order 9 delivered by delivery person 4
Order 0 delivered by delivery person 0
Order 3 delivered by delivery person 2
Order 6 delivered by delivery person 1
Order 4 delivered by delivery person 2
Order 1 delivered by delivery person 0
Order 7 delivered by delivery person 1
Order 5 delivered by delivery person 4
Order 8 delivered by delivery person 1
Order 2 delivered by delivery person 0
10 new customers served by client PID: 1245

```

berry@DESKTOP-O92GAB6: /

```
Server: Order 4 has been delivered.
Server: Order 2 has been delivered.
Server: Order 7 has been delivered.
Server: Order 8 has been delivered.
berry@DESKTOP-O92GAB6:/mnt/c/Users/lafci/Desktop/C
al$ ./HungryVeryMuch 8080 10 30 20
Connected to server...
Client PID: 1245
Order 0 placed at (23, 13)
Order 1 placed at (14, 16)
Order 2 placed at (4, 18)
Order 3 placed at (19, 19)
Order 4 placed at (9, 3)
Order 5 placed at (20, 17)
Order 6 placed at (15, 10)
Order 7 placed at (15, 18)
Order 8 placed at (11, 15)
Order 9 placed at (14, 2)
Server: Order 9 has been delivered.
Server: Order 3 has been delivered.
Server: Order 0 has been delivered.
Server: Order 6 has been delivered.
Server: Order 4 has been delivered.
Server: Order 1 has been delivered.
Server: Order 7 has been delivered.
Server: Order 5 has been delivered.
Server: Order 2 has been delivered.
Server: Order 8 has been delivered.
berry@DESKTOP-O92GAB6:/mnt/c/Users/lafci/Desktop/C
```



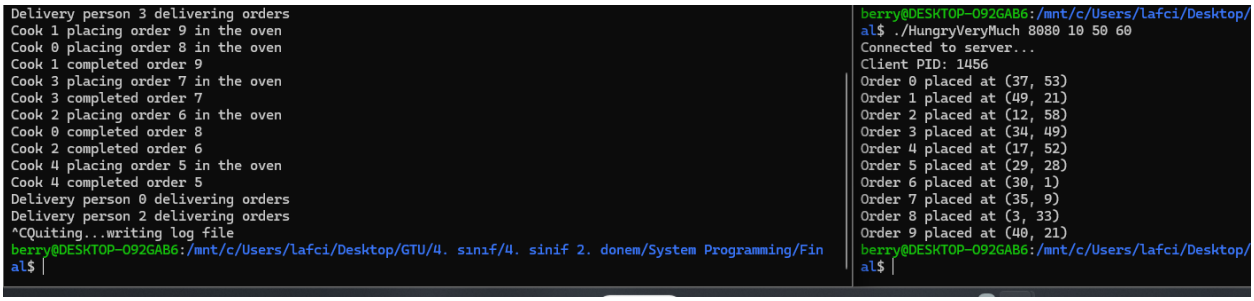
9-) Showing Pseudo inverse time(preparation and cook):



10-) Showing delivery time with Euclidean distance:

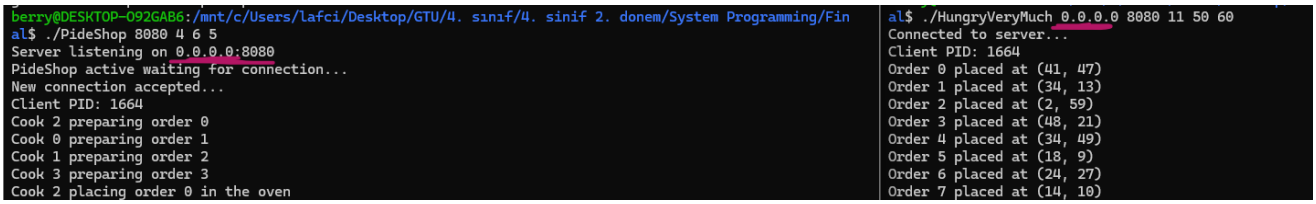
```
Delivery person 2 delivering orders
Delivery person 1 delivering orders
Delivery time 5.22
Order 9 delivered by delivery person 2
Delivery time 10.02
Order 3 delivered by delivery person 4
Delivery time 11.48
Order 0 delivered by delivery person 0
Delivery time 10.00
Order 6 delivered by delivery person 1
Delivery time 7.79
Order 5 delivered by delivery person 2
Delivery time 11.51
Order 4 delivered by delivery person 4
Delivery time 11.93
Order 1 delivered by delivery person 0
Delivery time 11.93
Order 7 delivered by delivery person 1
Delivery time 5.00
Order 2 delivered by delivery person 0
Delivery time 5.00
Order 8 delivered by delivery person 1
Done serving client pid: 1424
PideShop active waiting for connection...
```

11-) Ctrlc to server also closes the client:



12-) Ctrlc to client stops the order in server:

13-) With ip address given:



14-) Promoted delivery person: