Operating Systems 2018-B
Assignment 3:

Deadline: Monday, 23/04/2018

This assignment should be submitted as a *.tar.gz* archive, with C++ and README files inside. The archive should not contain generated object or executable files.

The assignment should be submitted in groups of 2 students. Both student should submit in Moodle. Names and IDs of the students in the pair must be written inside a README file in the archive, and *also* listed in the submission comment in Moodle.

Here is how you can create a flat submission archive of directory ex3 that contains files ex3/README, ex3/ex3.cpp, ex3/ex3.h, and so forth:

```
tar -C ex3 -czvf ex3.tar.gz .
```

Your assignment must compile without errors *or warnings* in a fully updated 64-bit Ubuntu 16.04 LTS distribution using the following command:

```
g++ -o ex3 *.cpp -std=c++11 -Wall -Wno-vla -pedantic -march=core2 -Os -pipe
    -fstack-protector-all -g3 -Wl,-O,1,-z,combreloc
```

Questions about the assignment should be asked in Piazza. Late submissions are penalized 10 points per day, and assignments can be submitted at most 5 days late. Special requests should be directed to your lab TA.

Introduction

A new fast food restaurant in Tel Aviv would like you to create a simulation program of its work. It will have a new interactive menu. A number of waiters work in the restaurant. Customers can "come" to the restaurant and check the list of dishes, and order a certain amount of the item they want.

Simulation

In this assignment, you will implement a program which simulates the problem described above. Each waiter and each customer will be simulated by a separate process. The different dishes will be listed in shared memory so all processes can access them. The entire simulation must use one or more sections of *shmem*.

1. Main process (the manager)
    a. Fill out the menu 5-7 items: *Id*, *Name* up to 15 chars, *Price* up to 100 NIS, *TotalOrdered* initialized to 0, and print it.
    b. Create "orders board" for orders (one for each customer): *CustomerId*, *ItemId*, *Amount*, *Done* as binary value initialize to *true* (to wait for orders).
    c. The main process should create any needed semaphores and start all sub processes. Any sub process each type {Customer or Waiter} should have an index according to the creation order.
    d. At the end of simulation time, main process should wait for termination of all sub processes, and print general information about the simulation: the total counts of each dish ordered, the total income of the restaurant during the simulation.

2. Customer process:
    a. If not elapsed simulation time
    b. Sleep for 3 to 6 seconds, randomly
    c. Read the menu (1 seconds)
    d. If the previous order has not yet been done, loop to (a)
    e. With the probability 0.5, the customer will order as follows:
        i. Randomly choose an item, randomly choose an amount (between 1 and 4);
        ii. Write the order to the "orders board" under customer index and set value of *Done* to *false*.
    f. With the probability 0.5, the customer does not order!
    g. Loop to (a)
3. Waiter process:
    a. If not elapsed simulation time
    b. Sleep for 1 to 2 seconds, randomly
    c. Read an order from the "order board"
    d. If there are no orders (*Done* is *true*), loop to (a)
    e. If there is row that isn't *Done* (value is *false*):
        i. Add the amount ordered to the totals for the item in main menu
        ii. Mark the order as *Done* (set to *true*)
    f. Loop to (a)

When a process is writing to a shared memory it must have unique access, with no other writers and no readers. This is the classic readers-writers problem.[1] Synchronization between waiters and customers should be done with the help of semaphores.

When starting the program, it should get the following arguments on the command line:

- Number of different dishes (up to 10)
- Number of waiters (up to 3)
- Number of customers (up to 10)

"Random" sleep times (a floating-point value):

- Time between customers reading the menu is 3 to 6 seconds
- Time between waiters checking orders is 1 to 2 seconds
- Total time running of simulation, less than 30 seconds

Every process should print to *stdout* if it reads the menu, or if it orders. The message should contain time since start of simulation, message about the event, the customer *or* waiter number and PID of the process. All error messages should be written to *stderr*.

Notes:

- If there was an error in the input parameters you should correctly terminate the program and write the reason for the error.
- Use a shared memory for the menu (items 1(a) and 1(b)).
- Using a shared resource, such as display to output: make sure that your process has unique access to the output (use a semaphore to protect all output).

[1] See https://en.wikipedia.org/wiki/Readers–writers problem

- Access the shared memory properly, according to the reader-writer algorithm (multiple readers are allowed, writers are allowed only one at a time with no reads). Use semaphores to implement this algorithm.
  YOU MUST IMPLEMENT THIS ALGORITHM AT LEAST ONCE IN YOUR SOLUTION TO RECEIVE FULL CREDIT

Optimization and Development

You could try to implement first the idea with 1 customer and 1 waiter.

After this runs correctly, create multiple customers. Then create also multiple waiters. Next make sure the readers/writers algorithm is implemented. Finally, add the reports at the end of the run.

Index	*Name*	*Price*	*TotalOrders*
0	Pizza	10.00	0
1	Salad	7.50	0
2	Hamburger	12.00	0
3	Spaghetti	9.00	0
4	Pie	9.50	0
5	Milkshake	6.00	0

Example Menu table

CustomerId	*ItemId*	*Amount*	*Done*
0	1	2	Yes
1	4	3	No
2	3	1	No

Example Orders board (For 3 customers)

Example 1

```
VirtualBox:~/OpSystem/ex3$ ./ex3 15
Input arguments are not valid!
```

Example 2

```
VirtualBox:~/OpSystem/ex3$ ./ex3 15 6 2 1
=====Simulation arguments=====
Simulation time: 15
Menu items count: 6
Customers count: 2
Waiters count: 1
==============================
 0.000 Main process ID 21579 start
==========Menu list===========
Id Name      Price  Orders
1  Pizza     10.00  0
2  Salad     8.00   0
3  Hamburger 12.00  0
4  Spaghetti 9.00   0
5  Pie       9.50   0
6  Milkshake 6.00   0
==============================
 0.000 Main process start creating sub-process
 0.000 Waiter 0: created PID 21582 PPID 21579
 0.000 Customer 1: created PID 21581 PPID 21579
 0.000 Customer 0: created PID 21580 PPID 21579
 3.038 Customer ID 1: reads a menu about Salad (ordered, amount 2)
 3.901 Customer ID 0: reads a menu about Milkshake (ordered, amount 1)
 4.899 Waiter ID 0: performs the order of customer ID 0 (1 Milkshake)
 6.272 Waiter ID 0: performs the order of customer ID 1 (2 Salad)
 7.066 Customer ID 1: reads a menu about Hamburger (doesn't want to order)
 7.889 Customer ID 0: reads a menu about Pie (ordered, amount 3)
 8.263 Waiter ID 0: performs the order of customer ID 0 (3 Pie)
11.252 Customer ID 1: reads a menu about Salad (ordered, amount 4)
12.237 Waiter ID 0: performs the order of customer ID 1 (4 Salad)
12.665 Customer ID 0: reads a menu about Pizza (doesn't want to order)
16.879 Customer ID 1: reads a menu about Hamburger (doesn't want to order)
16.879 Customer ID 1: PID 21581 end work PPID 21579
17.463 Customer ID 0: reads a menu about Spaghetti (doesn't want to order)
17.463 Customer ID 0: PID 21580 end work PPID 21579
17.519 Waiter ID 0: PID 21582 end work PPID 21579
==========Menu list===========
Id Name      Price  Orders
1  Pizza     10.00  0
2  Salad     8.00   6
3  Hamburger 12.00  0
4  Spaghetti 9.00   0
5  Pie       9.50   3
6  Milkshake 6.00   1
==============================
Total orders 10, for an amount 82.50 NIL
17.519 Main ID 21579 end work
17.519 End of simulation
```

Example 3

```
VirtualBox:~/OpSystem/ex3$ ./ex3 15 6 6 2
=====Simulation arguments=====
Simulation time: 15
Menu items count: 6
Customers count: 6
Waiters count: 2
==============================
 0.000 Main process ID 21504 start
==========Menu list===========
Id Name      Price  Orders
1  Pizza     10.00  0
2  Salad     8.00   0
3  Hamburger 12.00  0
4  Spaghetti 9.00   0
5  Pie       9.50   0
6  Milkshake 6.00   0
==============================
 0.000 Main process start creating sub-process
 0.000 Waiter 0: created PID 21511 PPID 21504
 0.000 Customer 5: created PID 21510 PPID 21504
 0.000 Customer 4: created PID 21509 PPID 21504
 0.000 Waiter 1: created PID 21512 PPID 21504
 0.000 Customer 3: created PID 21508 PPID 21504
 0.001 Customer 2: created PID 21507 PPID 21504
 0.001 Customer 1: created PID 21506 PPID 21504
 0.001 Customer 0: created PID 21505 PPID 21504
 3.941 Customer ID 1: reads a menu about Salad (ordered, amount 2)
 4.030 Waiter ID 0: performs the order of customer ID 1 (2 Salad)
 4.887 Customer ID 2: reads a menu about Hamburger (ordered, amount 1)
 4.979 Waiter ID 1: performs the order of customer ID 2 (1 Hamburger)
 5.036 Customer ID 4: reads a menu about Pie (doesn't want to order)
 5.552 Customer ID 0: reads a menu about Salad (doesn't want to order)
 5.762 Customer ID 3: reads a menu about Pie (ordered, amount 4)
 5.865 Waiter ID 0: performs the order of customer ID 3 (4 Pie)
 5.903 Customer ID 5: reads a menu about Hamburger (ordered, amount 2)
 6.543 Waiter ID 1: performs the order of customer ID 5 (2 Hamburger)
 9.282 Customer ID 1: reads a menu about Pizza (doesn't want to order)
10.355 Customer ID 5: reads a menu about Hamburger (ordered, amount 1)
10.517 Customer ID 2: reads a menu about Salad (doesn't want to order)
```

```
10.830 Customer ID 4: reads a menu about Salad (doesn't want to order)
11.319 Customer ID 0: reads a menu about Hamburger (doesn't want to order)
11.508 Customer ID 3: reads a menu about Pie (doesn't want to order)
14.682 Customer ID 0: reads a menu about Milkshake (doesn't want to order)
14.754 Customer ID 1: reads a menu about Hamburger (ordered, amount 2)
14.973 Customer ID 2: reads a menu about Pie (ordered, amount 1)
15.015 Waiter ID 1: performs the order of customer ID 1 (2 Hamburger)
15.136 Waiter ID 0: performs the order of customer ID 2 (1 Pie)
15.275 Customer ID 4: reads a menu about Pizza (ordered, amount 2)
15.275 Customer ID 4: PID 21509 end work PPID 21504
15.483 Customer ID 3: reads a menu about Milkshake (ordered, amount 2)
15.483 Customer ID 3: PID 21508 end work PPID 21504
15.871 Customer ID 5: reads a menu about Salad (ordered, amount 2)
15.871 Customer ID 5: PID 21510 end work PPID 21504
16.234 Waiter ID 1: performs the order of customer ID 3 (2 Milkshake)
17.019 Waiter ID 0: performs the order of customer ID 4 (2 Pizza)
17.611 Waiter ID 1: performs the order of customer ID 5 (2 Salad)
18.425 Customer ID 0: reads a menu about Milkshake (doesn't want to order)
18.425 Customer ID 0: PID 21505 end work PPID 21504
19.507 Customer ID 2: reads a menu about Salad (ordered, amount 4)
19.507 Customer ID 2: PID 21507 end work PPID 21504
19.586 Waiter ID 0: performs the order of customer ID 2 (4 Salad)
19.822 Customer ID 1: reads a menu about Milkshake (ordered, amount 4)
19.822 Customer ID 1: PID 21506 end work PPID 21504
20.694 Waiter ID 0: performs the order of customer ID 1 (4 Milkshake)
20.694 Waiter ID 0: PID 21511 end work PPID 21504
21.075 Waiter ID 1: PID 21512 end work PPID 21504
==========Menu list===========
Id Name      Price  Orders
1  Pizza     10.00  2
2  Salad     8.00   8
3  Hamburger 12.00  6
4  Spaghetti 9.00   0
5  Pie       9.50   5
6  Milkshake 6.00   6
==============================
Total orders 27, for an amount 239.50 NIL
21.076 Main ID 21504 end work
21.076 End of simulation
```