Homework #8

For the diagrams and written questions required for this homework assignment, upload a PDF file named `hw7.pdf` on Blackboard.

Question 1 (3 pt.)

Write a program that implements the dining philosophers problem in such a way that a deadlock situation is guaranteed not to occur. Represent the resource dependency diagram for your implementation, and justify why deadlock is avoided. Upload your code in a file named `q1.c`.

Question 2 (7 pt.)

A cluster of computers uses an interconnection network to allow its nodes to communicate with each other by sending packets through it. Consider a network formed of 16 nodes organized as a 4×4 grid (formally known as a 2D *mesh* topology), where the node on the first row and first column is referred to as node 0/0, the node on the first row and second column is referred to as node 0/1, etc.

Write a multi-threaded simulator for this network that continually injects packets to it from a random source into a random destination, and tracks their progress through the network. The network is modeled as a 2-dimensional array of mutexes, where the presence of a packet in a node is modeled by locking the mutex in the corresponding position on the matrix. In order for a packet to move to the next node, it must secure the next position (lock the next mutex) before it leaves its current position (unlocks the current mutex).

Let us assume that the network accepts two in-flight packets at once. This means that the parent thread should spawn two child threads, each running an infinite loop that produces a new packet and leads it to its destination. In each iteration of the loop, each child thread has the following tasks:

- Calculate a random source node to send a packet to the network (x_{src}/y_{src}).
- Calculate a random destination node to send the packet to (x_{dest}/y_{dest}), making sure that these coordinates are different from the source node.
- Packet occupies initial position x_{src}/y_{src}.
- While the packet has not reached its destination:
    - Calculate next position, where the packet can only jump 1 node up/down/right/left in one single coordinate at a time. Always choose the X coordinate first, while it differs. Move the packet in the Y coordinate only once the X coordinate matches the packet destination.
    - Release previous position.
- Packet releases position x_{dest}/y_{dest}.

a) Write a working implementation for the network simulator, which displays the coordinates of the current position for each packet at all times, as well as the next position that the packet is trying to access. Upload your code in a file named `q2.c`.

b) Run your code, and show how it eventually reaches a deadlock situation. Justify why this deadlock occurs, and copy the last part of your program output that illustrates the deadlock state.

c) Represent the deadlock state graphically in a grid, drawing the current position of the in-flight packets, as well as the next position they are attempting to reach. Show the cyclic resource dependency that leads to the deadlock.

d) What restriction would you add to the definition of this system to guarantee that no deadlock situation is possible? You can choose any kind of restriction, regarding the grid size, the way the packets are routed, the coordinates of the source and destination positions of a packet, etc.