1 Semaphores

The Hamburg airport has two runways that cross in some point E (see figure below). Arriving airplanes either choose runway AB or runway CD for landing.

In order to prevent two planes from crashing, synchronisation is inevitable. Therefore, the airport administration designs the following rules:

I. No two airplanes can use the same runway at the same time.
II. If an airplane passed point E on its runway, the other runway is open for landing.

You are allowed to use as many binary semaphores as you like in as many of the points A, B, C, D, or E as needed. Give a valid solution for both AB and CD airplanes with the correct usage of P( ) and V( ) operations for your semaphores! Discuss if your solution is fair!

2 Dining Philosophers Problem

Five philosophers are sitting around a table, doing one of two things: thinking or eating. While thinking, they are not eating, and while eating, they are not thinking. In the centre of the table is placed a big bowl with spaghetti. Furthermore, in front of each philosopher lies a plate and between each plate lies one chopstick. Thus, each philosopher has one chopstick to his left side and one chopstick to his right side. For eating spaghetti, a philosopher is assumed to use two chopsticks, but he can only use the two chopsticks on his immediate left and right side (see figure).
As the philosophers never speak to each other, a synchronisation problem arises due to the shared usage of resources (i.e. the chopsticks). Obviously, this scenario could easily lead to a deadlock where no philosopher is able to eat or to a lifelock where at least some philosophers have to starve (i.e. never manage to get two chopsticks at the same time while others do).

Give a solution for the above synchronisation problem by using lock variables, semaphores, or monitors and discuss your solution!

3 Deadlocks

In a multicore system 4 processes (threads) $P_i$, $i \in \{1, 2, 3, 4\}$, are competing for 2 resources $R_1$ and $R_2$. Both resources are available two times. In order to finish their computations, the processes have the following request (in that order).

\[ P_1 \rightarrow R_1, \quad P_3 \rightarrow R_2, \quad P_4 \rightarrow R_2, \quad P_2 \rightarrow R_1, \quad P_1 \rightarrow R_2, \quad P_3 \rightarrow R_1, \quad P_2 \rightarrow R_2 \]

Assuming that all resources are only exclusively useable, resources cannot be withdrawn from a process, and processes do not release assigned resources while waiting for the allocation of other resources, the above situation might lead to a deadlock.

Draw an allocation/request graph for this scenario and discuss if the above situation is deadlock-free and in what ordering of execution – if possible – processes $P_1$ to $P_4$ might finish their computations!