Exercise 4 - Synchronization with Semaphores

Question 1: Semaphore-Operations

Figure 1.1 shows the sequence of semaphore operations at the beginning and at the end of the tasks A, B, C.
Determine for the 4 cases a,b,c,d, given in table 1.1, whether and in which sequence the tasks are executed, using the initializations of the semaphore variables given in table 1.1.

Task A	**Task B**	**Task C**
P(SA)	**P(SB)**	**P(SC)**
P(SA)	**.**	**P(SC)**
P(SA)	**.**	**P(SC)**
.	**.**	**.**
.		**.**
.	**V(SC)**	**V(SB)**
V(SB)	**V(SA)**	**V(SB)**
END	**END**	**END**

Figure 1.1: Sequence of semaphore operations

variable	a)	b)	c)	d)
SA	2	3	2	0
SB	0	0	1	0
SC	2	2	1	3

Table 1.1: Initial values of the semaphore variables

Question 2: Synchronization with Semaphores

A production system, consisting of 3 subprocesses, is to be automated by a program which contains the following 4 tasks:

- **SUPERVISION1, SUPERVISION2, SUPERVISION3**
  to read in data from the respective subprocesses.
- **PROTOCOL**
  to output the measured values in tables and diagrams on a printer.

The following sequence of the tasks must be created by a suitable synchronization with semaphores:

- The **PROTOCOL** -task should be executed after each supervision-task.
- The sequence of the supervision-tasks is given by the number in their name. **SUPERVISION1** is therefore the first task.
- The sequence given above should be repeated cyclically.

The program of each task should be executed completely, before another task can be started.

Questions:

a) Write down the names of the tasks in the desired sequence.

b) Insert into each task the semaphore-operations required to guarantee the desired task sequence.

c) With which values do the semaphore variables have to be initialized?

Question 3: Semaphore-Operations

The two tasks **NEWVALUE** and **CALCULATE** access a data field called **QUEUE** (see Figure 3.1):

- The task **NEWVALUE** does a write access (procedure **WRITE**).
- The task **CALCULATE** does a read access (procedure **READ**).

a) Insert in Figure 3.1 suitable semaphore operations by which a simultaneous access is excluded. Give the initial values of the semaphore variables. Which kind of synchronization is this ?

b) Additionally, make sure that a certain number of read accesses can be executed only if the corresponding number of write accesses has been executed. Add to figure 4.4.1 the necessary semaphore operations and give the initial values of the semaphore variables.

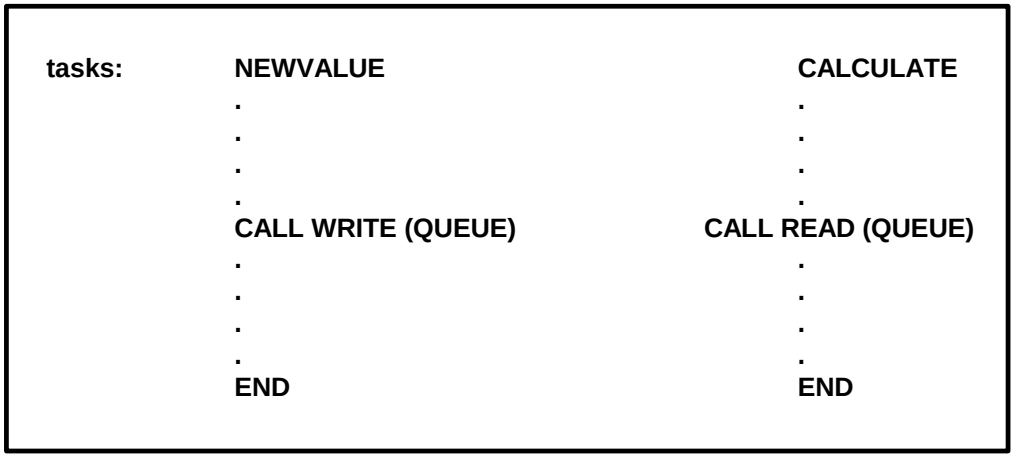

*Figure 3.1: Synchronization of the access of the data field **QUEUE***

Question 4:Synchronization with Semaphores

The Priority-Inversion Problem

In this question we want to look at the priority-inversion problem for the synchronization of processes with semaphores. In order to illustrate the problem, we have a look at a simple scenario first.

Scenario 1:
Let A, B, and C be three tasks with descending priorities, where A has the highest priority (1). The tasks A and C use a common resource (e.g. a memory area). This resource is protected against simultaneous access with a semaphore. Figure 4.1 shows the structure of the three tasks of scenario 1. The respective execution times are written next to the dotted lines (program code).

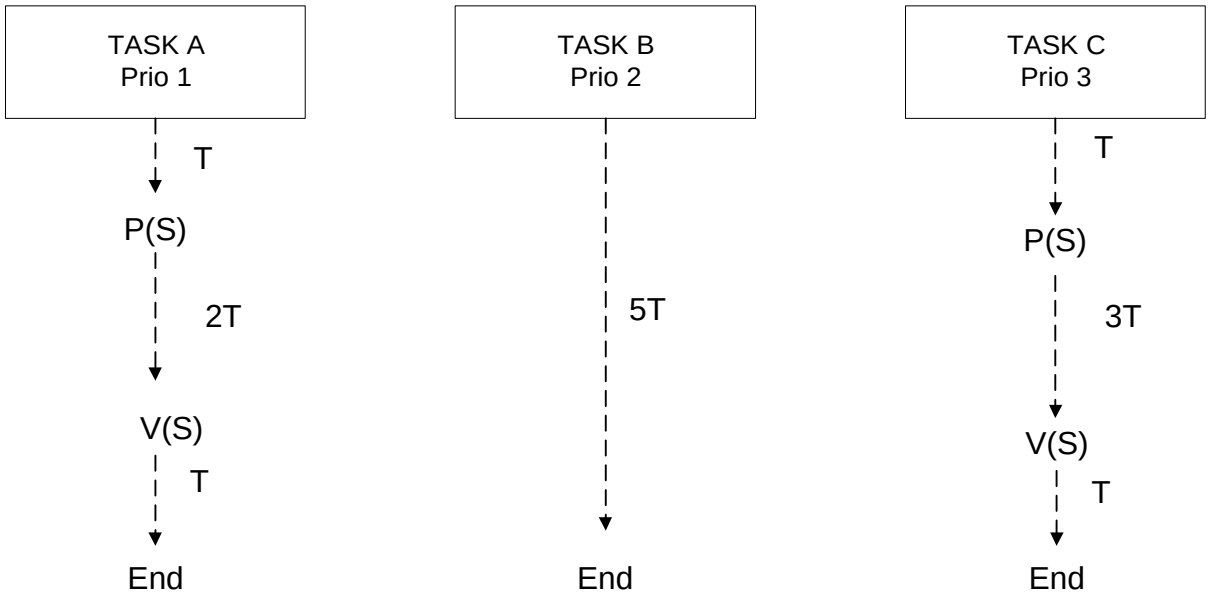

Figure 4.1: Structure of the tasks of scenario 1

The tasks are activated (only once) at the following times:

task A: t = 2T
task B: t = 4T
task C: t = 0T

Remark:
In this question, we assume preemptive scheduling with fixed priorities. The initial values of all semaphore variables are 1.

a) Draw in figure 4.3 the desired execution sequence for the time interval $0 \leq t \leq 14T$. (The state „idle“ is not to be depicted in the diagram). Draw in figure 4.4 the actual execution sequence with the according task states of the three tasks for the time interval $0 \leq t \leq 14T$.

b) Let us assume, that there would be further processes with priorities between those of A and C which do not access the common memory area (no semaphore operations). The

individual priorities and execution times are unknown. How long would then the high priority task A be delayed in the worst case?

To avoid this problem, a new strategy will be used in the following. As soon as a task X blocks a task with a higher priority (unsuccessful P-operation of a task with a higher priority), the task ignores its own priority and executes the *critical section* (the section between a P- and a V-operation) with the highest priority of all tasks that are blocked by it. As the task ‚inherits' the priority of another task with a higher priority, this synchronization method is called the '**Priority Inheritance Protocol**'.

c) Draw in Figure 4.5 the new actual execution sequence.

d) Using this method, how long can a task with high priority be blocked maximally by a task with lower priority ? (compared to the simple solution with semaphores)

Scenario 2:
Three processes A, B, and C with different priorities (A - highest, B - medium, C - lowest) are given. The three processes are synchronized with three binary[1] semaphores S0, S1, S2. The semaphores S1 and S2 guarantee the exclusive access on two common memory areas. Figure 4.2 shows the structure of the three tasks A, B, and C of scenario 2.

The tasks are activated (only once) at the following times:

task A: t = 5T
task B: t = 2T
task C: t = 0T

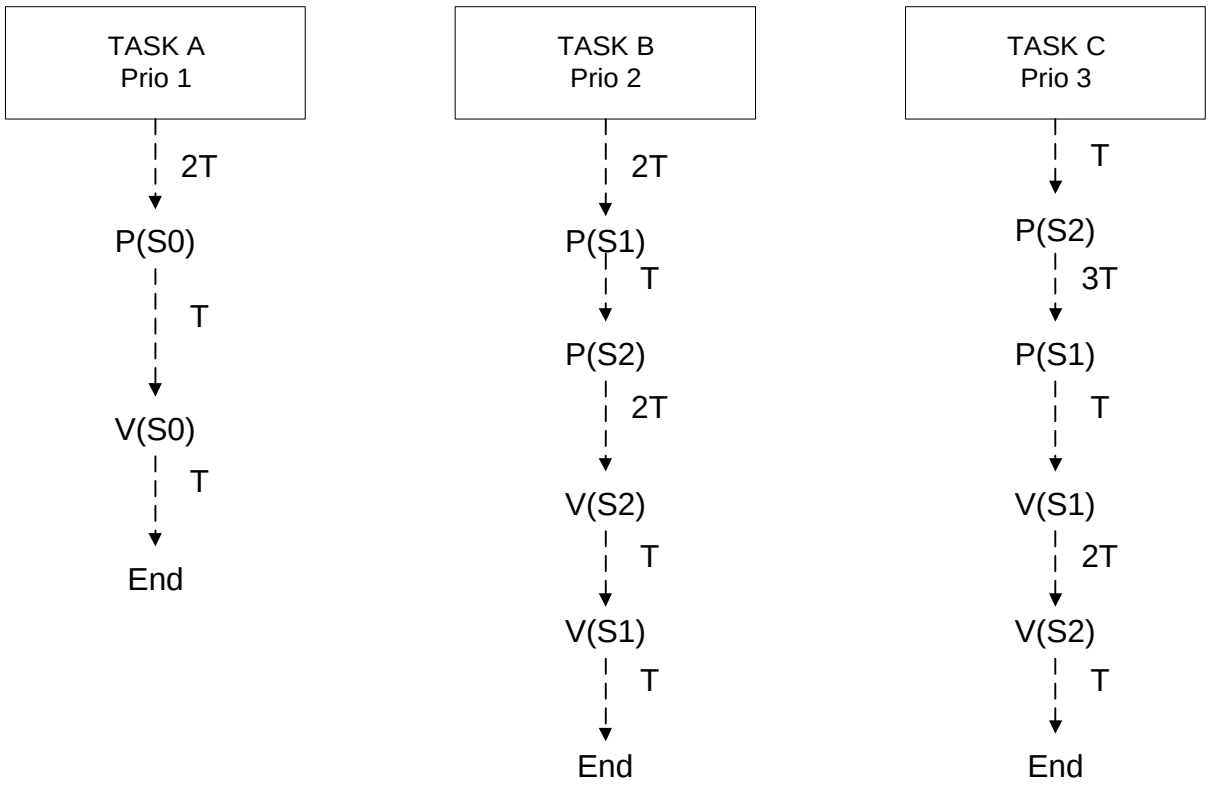

Figure 4.2: Structure of the tasks A, B, and C of scenario 2

[1]A binary semaphore can only have the values 0 and 1

e) Draw in Figure 4.6 the desired execution sequence for the time interval $0 \leq t \leq 19T$. Draw in Figure 4.7 the actual execution sequence and the task states resulting from the strategy of question c) (Priority Inheritance Protocol) for the time interval $0 \leq t \leq 19T$. What can you notice?

The Priority Ceiling Protocol

Finally, we look at an extension of the 'Priority Inheritance Protocol' – the so called 'Priority Ceiling Protocol'. An objective of this protocol is to avoid deadlocks.

The basic idea of this protocol is to guarantee the following: If a task X interrupts the critical section of another task and executes its own critical section then the priority with which this new critical section is executed is higher than all inherited priorities of all interrupted critical sections.
For this purpose, an upper priority bound is assigned to every semaphore (Priority Ceiling). This upper bound is the priority of the task with the highest priority, that uses this semaphore.
A task X is permitted to enter a new critical section (P-operation) only if the priority of task X is higher than all priority bounds of all semaphores that are used at the moment by other tasks (semaphore-variable = 0).

Example: A semaphore S is used by two tasks with the priorities 2 and 4. It gets the priority bound 2 (highest priority).
If S (as the only semaphore) is used by a task and another task tries to execute a P-operation then this operation will be executed only if the priority of this task is higher than 2.

f) Which upper priority bounds are assigned to the semaphores S0, S1 and S2 in scenario 2 ?

g) Draw in Figure 4.8 the actual execution sequence and the task states for the time interval $0 \leq t \leq 19T$ resulting if the Priority Ceiling Protocol is applied.

Question 5: Rate-monotonic Scheduling

Rate-monotonic scheduling algorithm:

The priority of a process is determined by its period time. The highest priority is assigned to the process with the shortest period time. The remaining processes get priorities according to the ascending order of their period times.

Theorem of Liu and Layland (schedulability test):

A set of n *independent* (i.e. no shared resources, no synchronization) periodic tasks which has been scheduled according to the rate-monotonic algorithm guarantees all deadlines if:

$$\frac{C1}{T1} + \frac{C2}{T2} + ... + \frac{Cn}{Tn} \leq U(n) = n \cdot (2^{1/n} - 1)$$

with:
Ci = worst-case execution time of task i
Ti = period time of task i
U(n) = utilization bound for n tasks

problem:
Interactions and access on shared resources are not considered in this theorem.
⌋ Extension with task synchronization. But this can cause blockings and deadlocks.
Therefor the blocking times must be considered, as well.

Schedulability test for a task set with mutual blocking

Step 1: Calculation of the total processor utilization for all tasks

$$U_{total} = \sum_{i=1}^{n} \frac{C_i}{T_i}$$

Step 2: Calculation of the maximum blocking time

$$B_{total} = \max_{i=1}^{n} \left(\frac{B_i}{T_i} \right)$$ with: Bi = blocking time of task i

Step 3: Calculation of the processor utilization bound

$$U(n) = n \cdot (2^{1/n} - 1)$$

Step 4: Comparison of the effective processor utilization and the processor utilization bound

$$U_{total} + B_{total} \leq U(n)$$

Questions:

Given is a set of computation processes represented in Figure 5.1. The synchronization of the access of the processes on the resources is done using semaphores and the "priority ceiling protocol".

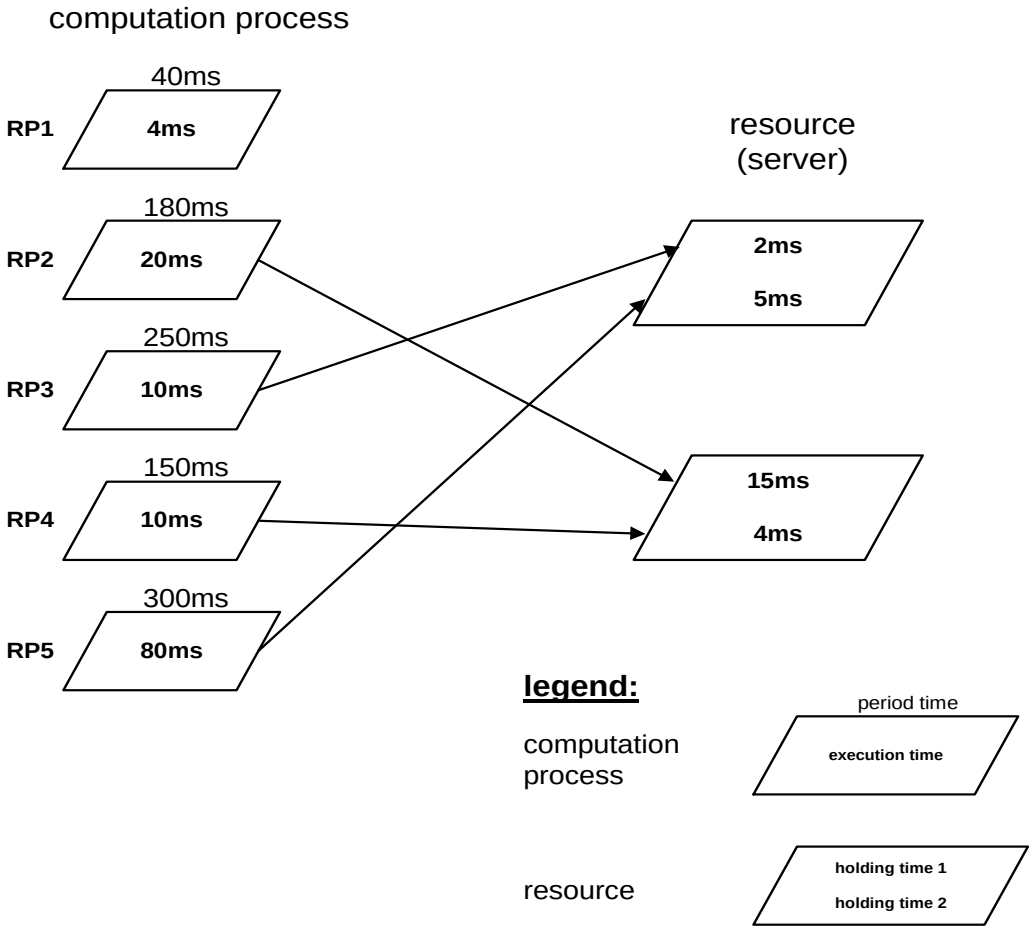

Figure 5.1

a) Write down in a table the execution time, the period time and the blocking time of each computation process. Assign the priorities to the computation processes according to the rate-monotonic algorithm.

b) Calculate the total processor utilization of all computation processes.

c) What is the maximum blocking time ?

d) Calculate the processor utilization bound.

e) Is the given set of computation processes schedulable ?

Solution sheet 1 for question 4

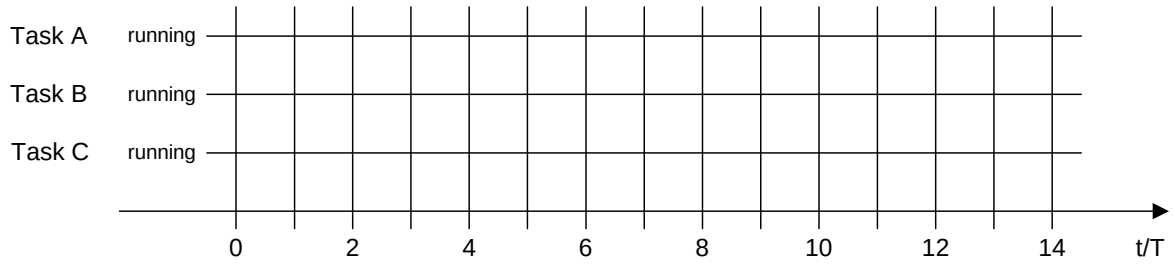

Figure 4.3: Desired execution sequence of the processes in scenario 1

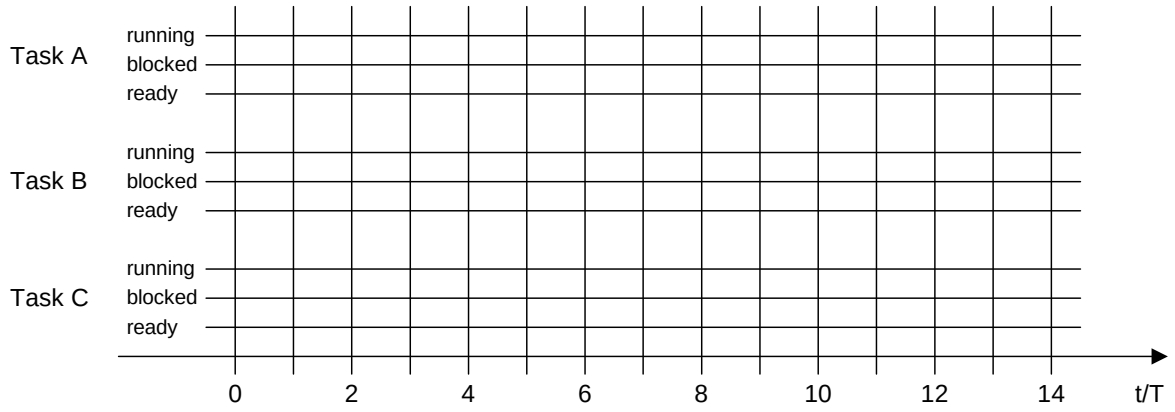

Figure 4.4: Actual execution sequence of the processes in question a)

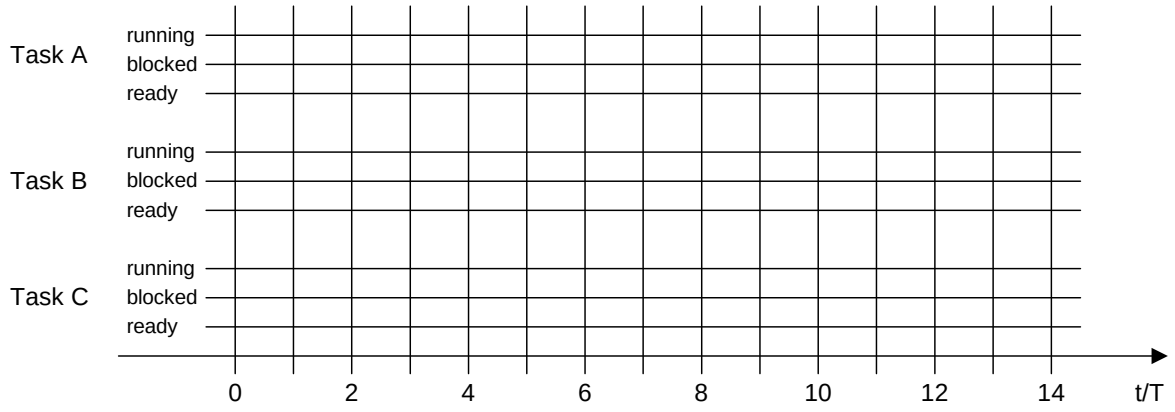

Figure 4.5: Actual execution sequence of the processes in question c)

Solution sheet 2 for question 4

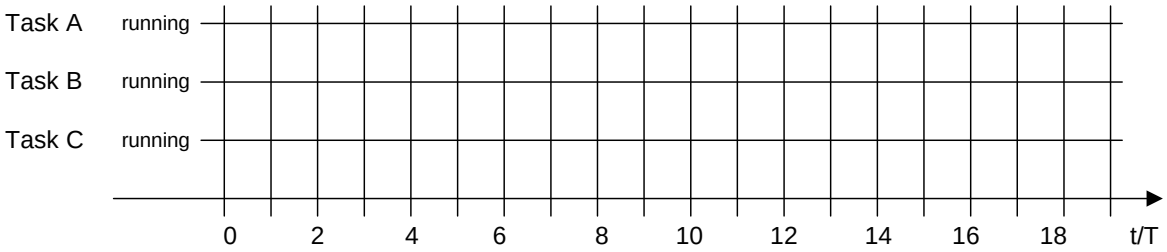

Figure 4.6: Desired execution sequence of the processes in scenario 2

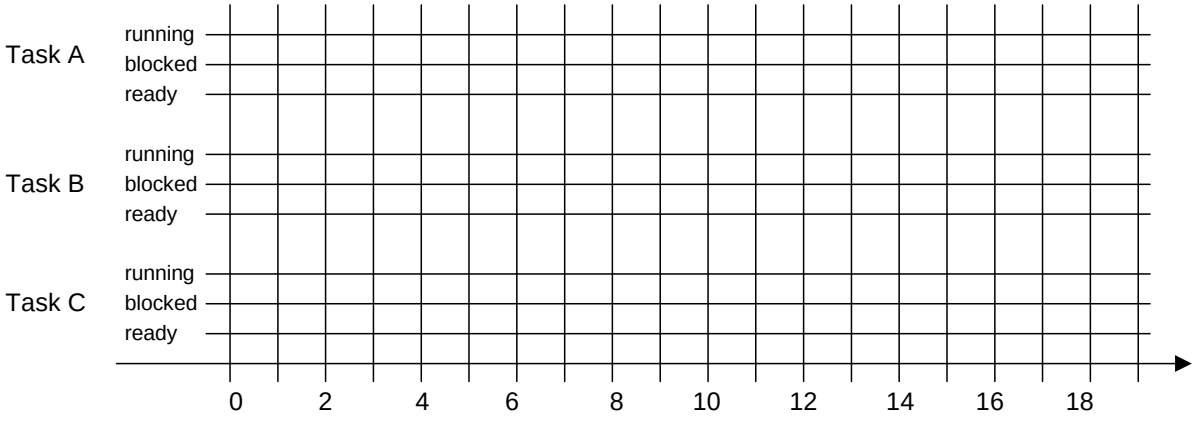

Figure 4.7: Actual execution sequence of the processes in question e)

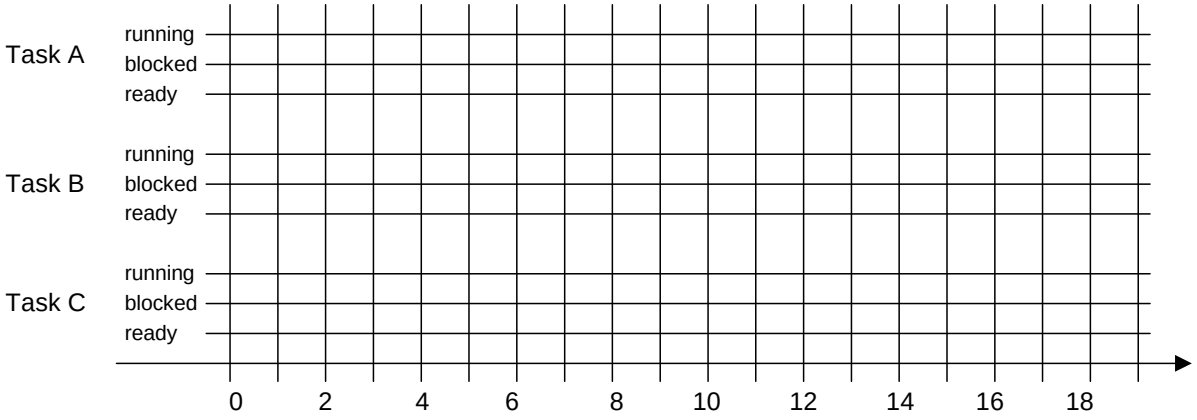

Figure 4.8: Actual execution sequence of the processes in question g)