# Sample exam questions, ppvt 2003

This is a collection of sample questions. Their solutions were discussed in class 4 Mar. The mock exam intended to give you a feeling for the design of the upcoming exam. Some of the questions here are taken from the exercises of the book, but that is only because they happened to be suitable. There is no implication that the exam will also use questions from the text book. The number and balance of question in the exam is not reflected below, only the style and range.

**Question 1.** Consider the following program:

```
co <await (x > 0) x = x - 1;>
// <await (x < 0) x = x + 2;>
// <await (x == 0) x = x - 1;>
oc
```

For what initial values of $x$ does the program terminate, assuming that scheduling is weakly fair? What are the corresponding values? Explain your solution.

**Question 2.** Consider the following program:

```
int x = 10, c = true;

co <await x == 0>; c = false
// while (c) <x = x - 1>;
oc
```

1. Will the program terminate if the scheduling is weakly fair? Explain.
2. Will the program terminate if the scheduling is strongly fair? Explain.

Add the following arm to the **co** statement:

```
while (c) { if (x < 0) <x = 10>; }
```

Repeat questions 1 and 2 above for this three-process program.

**Question 3.**

1. Consider the *f*etch and add (FA) instruction that increments a variable and returns its updated value. FA is atomic. Use this instruction to design an $n$-process barrier. (Hint: your solution need not taking efficiency into consideration)
2. Use the FA to implement a critical section for n processes.

**Question 4.** It is possible to implement a reusable $n$-process barrier with two semaphores and one counter. These are declared as follows:

```
int count = 0;
sem arrive = 1, go = 0;
```

Develop a solution. (Hint: Use the idea of passing the baton)

**Question 5.** Two processes executing the same program $A$ and $B$ are cooperating to solve a problem and need to exchange information. This is known as interacting peers. Assume that process $A$ contains some information $a$ and that process $B$ contains some information $b$. After the exchange both $A$ and $B$ should contain $a$ and $b$.

1. Solve this interaction problem using monitors.
2. Solve this interaction problem using protected types.

**Question 6.** Two kinds of processes, A's and B's enter the room. An A process cannot leave the room until it meets two B processes, and a B process cannot leave until it meets one A process. Each kind of processes leaves the room - without meeting any other processes - one it has met the required number of other processes.

1. Develop a server process to implement this synchronization.
2. Implement this synchronization without using a server.

**Question 7.** The *Stable Marriage Problem* is the following. Let **Man[1:n]** and **Woman[1:n]** be arrays of processes. Each man ranks the women from 1 to $n$, and each woman ranks the men from 1 to $n$. (A ranking is a permutation of the integers from 1 to $n$) A *pairing* is a one-to-one correspondence of men and women. A pairing is *stable* if, for two men **Man[i]** and **Man[j]** and their paired women **Woman[p]** and **Woman[q]**, both of the following conditions are satisfied.

1. **Man[i]** ranks **Woman[p]** higher than **Woman[q]**, or **Woman[q]** ranks **Man[j]** higher than **Man[i]**; and
2. **Man[j]** ranks **Woman[q]** higher than **Woman[p]**, or **Woman[p]** ranks **Man[i]** higher than **Man[j]**

Put differently, a pairing is unstable if a man and woman would both prefer each other to their current pair. A solution to the stable marriage problem is a set of pairings, all of which are stable.

Write a parallel program that simulates the stable marriage problem. You need not take termination of the program into account.

**Question 8.** Write a parallelized insertion sort using Linda. The tuples are either integers, representing the initial data to be sorted, or pairs, representing the sorted list in the following way:

$$[i, j, k] \Leftrightarrow (\bot, i), (i, j), (j, k), (k, \infty)$$

where $\bot$ represents the smallest integer and $\infty$ the largest.

1. Write a program that builds up the pairs of the list in parallel.
2. Write a program that writes out the sorted list.

How many Linda calls(IN, OUT, RD, INP, RDP, EVAL) do you need to sort a list of length $n$?