Operating Systems
Constructor University
Dr. Jürgen Schönwälder

Module: CO-562
Date: 2025-10-03
Due: 2025-10-10

Problem Sheet #5

Problem 5.1: *readers / writers problem* (1+1+1 = 3 points)

Below are three incorrect solutions of the readers-writers problem. Explain why the solutions works or in which situations the solutions fail to work correctly. The solutions use the following common definitions:

```
shared object data;
shared int readcount = 0;
semaphore mutex = 1, writer = 1;
```

a)
```
void reader()
{
    down(&mutex);
    readcount = readcount + 1;
    if (readcount == 1) down(&writer);
    up(&mutex);
    read_shared_object(&data);
    down(&mutex);
    readcount = readcount - 1;
    up(&mutex);
    if (readcount == 0) up(&writer);
}
```

```
void writer()
{
    down(&writer);
    write_shared_object(&data);
    up(&writer);
}
```

b)
```
void reader()
{
    down(&mutex);
    readcount = readcount + 1;
    if (readcount == 1) down(&writer);
    up(&mutex);
    read_shared_object(&data);
    down(&mutex);
    readcount = readcount - 1;
    if (readcount == 0) {
        up(&mutex);
        up(&writer);
    } else {
        up(&mutex);
    }
}
```

```
void writer()
{
    down(&writer);
    write_shared_object(&data);
    up(&writer);
}
```

c)
```
void reader()
{
    down(&mutex);
    readcount = readcount + 1;
    if (readcount == 1) down(&writer);
    up(&mutex);
    read_shared_object(&data);
    down(&mutex);
    readcount = readcount - 1;
    if (readcount == 0) up(&writer);
    up(&mutex);
}
```

```
void writer()
{
    down(&writer);
    down(&mutex);
    write_shared_object(&data);
    up(&mutex);
    up(&writer);
}
```

Problem 5.2: *good tasting coffee* (3 points)

You are given the following code fragment. Every function is running as an independent thread. Using the three semaphores `s1`, `s2`, `s3`, make sure that the program prints the message "the coffee tastes so good here". Add the missing semaphore operations and complete the semaphore initializations.

```
semaphore s1 =  ;
semaphore s2 =  ;
semaphore s3 =  ;

void a(void) {

   printf("the ");

   printf("tastes ");

   printf("good ");
}

void b(void) {

   printf("coffee ");

   printf("here");

}

void c(void) {

   printf("so ");

}
```

Problem 5.3: *bounded buffer with posix semaphores* (2+1+1 = 4 points)

We discussed a simple minded bounded buffer implementation that uses busy waiting and suffers from race conditions. The source code can be found on the gitlab server.

a) Rewrite the bounded buffer to avoid race conditions during the updates of the variables implementing the shared bounded buffer and to replace the busy waiting loops.

b) Explain the use of the atomic variables to generate a sequence of numbers. Which problem is addressed here?

c) Explain the use of atomic variables to implement a checker board for validating a likely correct execution of the concurrent program. Which problem is addressed here?