NOTE *These weekly exercises are individual (unless marked otherwise).*

READING *chapter 10 up to and incl. 10.3, 10.6, 10.7, 10.9.5*
(and for more technical details: 10.9.1, 10.9.2, 10.9.3)

HAND IN *Please hand in the exercises in PDF format in Brightspace.*

Exercise 1

Nicholas Nethercote is a developer working on Mozilla Firefox. His main interest is solving memory problems in Firefox. In a presentation given about the subject he explains the issues and solutions they use: `http://blog.mozilla.org/nnethercote/2012/01/17/`. Study the presentation and answer the following questions. **Use at most 200 words for each question**.

A Explain the concept of "slop" in your own words.

B In Mozilla's MemShrink project, how were the problems relating to "slop" discovered? Explain the used method in your own words.

C Is the slop internal or external fragmentation?

D Does the page and frame size have an influence on this problem?

Exercise 2

The exercise of this week is about memory usage/handling in a modern object oriented language, which can cause internal and external fragmentation. Assume the following properties of the hypothetical **'Radboud' Virtual Machine**:

Type	Memory	
boolean	1	byte
byte	1	byte
char	2	bytes
short	2	bytes
int	4	bytes
long	8	bytes
float	4	bytes
double	8	bytes

- The Virtual Machine is **32 bits**, so one word is 4 bytes.
- The memory consumption of the basic types is listed in the table. These **types should be aligned** to the size of the type (an int is 4 bytes aligned).
- **Objects** are always allocated on the heap. Objects start with a header consisting of one word garbage collect information, one word dynamic typing information, one word object monitor (for locking by the way of the synchronised statement). **Object headers needs to be aligned** on 8 byte boundaries.
- **Arrays** are never allocated inline (so always on the heap, like languages as Java). Arrays also have a header consisting of the standard object header and one int with the number of elements in the array, followed directly by the contents of the array, without any other overhead in the structure.

```
public class Padding {
  double a;
  int b;
  boolean b_disabled;
  short c;
  boolean c_enabled;
  double d;
  Basic* object1 = new Basic();
  double* lala = new double[7];
  Basic object2;
}
```

listing 1 Padding example.

```
public class Basic {
  double a;
  short b;
  byte c;
  double d;
  boolean e;
  Basic() {
    a = 37.0;
    b = 42;
    c = 255;
    d = 3.1418;
    e = true;
  }
  short calc() {
    return b;
  }
}
```

listing 2 Basic program.

An example program is listed in listings 1 and 2. Note: most (statically typed) programming languages have these padding issues (C, Objective-C, C++, Java, C#, etc).

A How much memory is used by both data structures when allocated?

B Optimise these data structures so they use less memory.

Exercise 3

What is the copy-on-write feature, and under what circumstances is its use beneficial? What hardware support is required to implement this feature?

Exercise 4

Consider a demand-paging system with the following time-measured utilizations:

CPU utilization	20%
Paging disk	97.7%
Other I/O devices	5%

For each of the following, indicate whether it will (or is likely to) improve CPU utilization. Explain your answers

A Install a faster CPU.

B Install a bigger paging disk.

C Increase the degree of multiprogramming.

D Decrease the degree of multiprogramming.

E Install more main memory

F Install a faster hard disk or multiple controllers with multiple hard disks.

G Add prepaging to the page-fetch algorithms.

H Increase the page size.