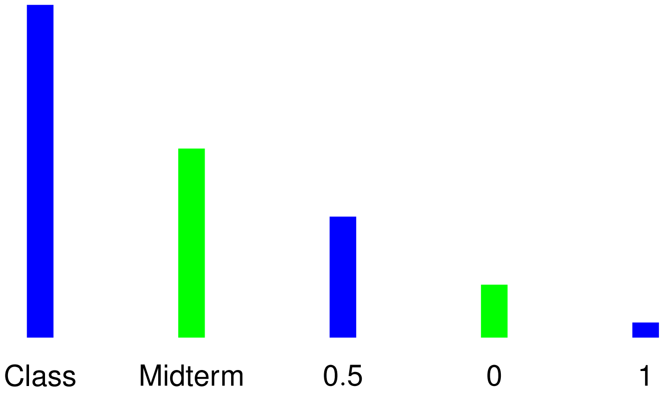
# The Midterm Exam: Comments & Solutions

Michał Kapałka

EPFL, LPD

STiDC'06, 23.I 2007

## Statistics



## Problem 1

**Task:** Implement Test&Set out of binary consensus objects and atomic registers.

## A Reminder

- Binary consensus is a consensus object that accepts only 0 and 1 as a proposed value.

```
propose(0 or 1)
```

- Test&Set maintains a binary value *x* (init. to 0), and has only one operation with the following sequential spec.:

```
operation test&set()
begin
  y := x;
  x := 1;
  return y;
end
```

## The Idea

- Test&Set $=$ first invocation returns 0, others return 1
- Concurrent invocations $\Rightarrow$ need to elect one process (winner) $\Rightarrow$ need to reach consensus on who the winner is

## An Algorithm (1)

**uses**: $C[1, \ldots, n]$ – binary consensus objects, $R[1, \ldots, n]$ – atomic SWMR registers

**initially**: $\mathit{firstInv}_i = \mathit{true}$ at every process $p_i$, $R[1, \ldots, n] = \mathit{false}$

## An Algorithm (2)

```
**upon** $\mathit{test\&set}_i()$ **do**
    **if** $\mathit{firstInv}_i = \mathit{false}$ **then return** *1*
    $\mathit{firstInv}_i \leftarrow \mathit{false}$
    $R[i].\mathit{write}(\mathit{true})$
    $v = 0,\ k \leftarrow 0$
    **while** $v = 0$ **do**
        $k \leftarrow k + 1$
        **if** $R[k].\mathit{read}() = \mathit{true}$ **then** $v \leftarrow 1$
        $v \leftarrow C[k].\mathit{propose}_i(v)$
    **if** $k = i$ **then return** *0*
    **return** *1*
```

## Problem 2 – Shared Object GetId

Shared object GetId maintains a value *lastId* (init. to 0) and an array of values *id*$[1, \ldots, n]$ (all init. to 0). It has one operation with the following sequential spec.:

```
operation getId()
begin
  if id[i] = 0 then
  begin
    lastId := lastId + 1;
    id[i] := lastId;
  end
  return id[i];
end
```

## Problem 2

**Question:** What is the consensus number of shared object GetId?

Consensus number $=$ the maximum number of processes, amongst which the object can implement consensus

**Answer:** 2

## The Idea

We need two steps:

1. The consensus number of GetId is at least 2
2. The consensus number of GetId is at most 2

## Step 1

The consensus number of GetId is at least 2 $\Rightarrow$ we can implement 2-consensus using GetId:

**uses**: $G$ – GetId object, $R[1,2]$ – array of atomic registers

```
upon propose_i(v_i) do
    $R[i] \leftarrow v_i$
    if $G.getId_i() = 1$ then return $v_i$
    return $R[3-i]$
```

## Step 2

The consensus number of GetId is at most 2 $\Rightarrow$ we can implement GetId using an object that is known to have consensus number 2, e.g., Test&Set:

**uses**: $T[1, \ldots, n]$ – array of Test&Set objects
**initially**: $id_i = \perp$ for every process $p_i$

```
upon getId_i() do
    if $id_i \neq \perp$ then return $id_i$
    $k \leftarrow 1$
    while $T[k].test\&set_i() = 1$ do $k \leftarrow k + 1$
    $id_i \leftarrow k$
    return $k$
```