Chalmers | GÖTEBORGS UNIVERSITET
K. V. S. Prasad, Computer Science and Engineering

Concurrent Programming TDA381/DIT390

Tuesday 23 October 2012, 14:00 to 18:00.

K. V. S. Prasad, tel. 0730 79 43 61

- Maximum you can score on the exam: 72 points. This paper has four pages, with seven questions, each carrying 12 points. Choose any six questions to answer. If you attempt all seven questions, we will ignore the question on which you score the least points.

  To pass the course, you need to pass each lab, and get at least 24 points on the exam. Further requirements for grades (Betygsgränser) are as follows, for total points on exam + labs:

  Chalmers: grade 3: 40 - 59 points, grade 4: 60 - 79 points, grade 5: 80 - 104 points.
  Chalmers ETCS: E = 40–47, D = 48–59, C = 60–75, B = 76–87, A = 88-104
  GU: Godkänd 45-79 points, Väl godkänd 80-104 points

- Results: within 21 days.

- **Permitted materials (Hjälpmedel):**
  - Dictionary (Ordlista/ordbok)

- **Notes: Please read these**
  - Time planning: you have 40 minutes for each of the six questions you will answer.
  - Start each question on a new page.
  - Answers in English only, please. Our graders do not read Swedish.
  - A SUMMARY follows of Ben-Ari's pseudo-code notation, used in this question paper.
  - Ben-Ari's pseudo-code should suffice for your programs, but you can use Java, JR, or Erlang if you think they are appropriate. The exact syntax of the programming notations you use is not so important as long as the graders can understand the intended meaning. If you are unsure, add an explanation of your notation.
  - If a question does not give you all the details you need, make reasonable assumptions, but state them clearly. If your solution works under certain conditions, state the conditions.
  - Be as precise as you can. Programs are mathematical objects, and discussions about them may be formal or informal, but are best mathematically argued. Handwaving arguments will get only partial credit. Unnecessarily complicated solutions will lose some points.
  - DON'T PANIC!

SUMMARY OF BEN-ARI'S PSEUDO-CODE NOTATION

Global variables are declared centred at the top of the program.

Data declarations are of the form `integer i := 1` or `boolean b := true`, giving type, variable name, and initial value, if any. Assignment is written `:=` also in executable statements. Arrays are declared giving the element type, the index range, the name of the array and the initial values. E.g., `integer array [1..n] counts := [0, ..., 0]`.

Next, the statements of the processes, often in two columns headed by the names of the processes. If several processes `p(i)` have the same code, parameterised by `i`, they are given in one column.

So in Question 1, *p* and *q* are processes that the main program runs in parallel. The declarations of $flag$ and *n* are global.

Numbered statements are atomic. If a continuation line is needed, it is left un-numbered or numbered by an underscore `p-`. Thus `loop forever`, `repeat` and so on are not numbered. Assignments and expression evaluations are atomic.

Indentation indicates the substatements of compound statements.

The synchronisation statement `await b` is equivalent to `while not b do nothing`. This may be literally true in machine level code, but at higher level, think of `await` as a sleeping version of the busy loop.

For channels, `ch => x` means the value of the message received from the channel `ch` is assigned to the variable `x`. and `ch <= x` means that the value of the variable `x` is sent on the channel `ch`.

When asked for a scenario, just list the labels of the statements in the order of execution.

——-END of SUMMARY——