Australian National University

CONFIRMING PREREQUISITES

First assignment for Real-Time and Embedded Systems 2019

Uwe R. Zimmer

This first assignment is meant to bring everybody on the same page in terms of Computer Science background and especially programming abilities. Read the provided examples with extreme care and be highly self-critical. Some of you will have seen all of this before and this first exercise will be a quick re-confirmation of your skills. Everybody else will need to spend a lot of time with those examples to make sure that you actually understand every word in them. What is tested in there in short is: A fourth year equivalent programming background and a solid understanding of concurrent systems. In more detail, the following list of programming concepts is explained by simple examples. The first chapter in the lectur also contains basic explanations of those concepts. It is suggested that you print this page and explicitly tick off every individual concept. Show this to your tutor.

- Separation between **Specification** (interface) and **implementation** (body) parts.
- **Variable** declarations (incl. **Constants** and **Initializations**).
- **Type system** with specific examples for numeric types (**modulo**, **ranges**), **enumeration** types, **records** as well as **type-attributes** and default **Initializations**. The example set is too small to provide an exhaustive set of types and data-structures.
- **Exceptions** incl. exception propagation.
- **Information hiding** as part of a modularization system.

All items up to here are exemplifying core computer science concepts as they are expected by a second year computer science student. If you are struggling with those concepts you have to see us straight away.

- **Generic** programming (polymorphism). This is a core programming concept which is available in different programming languages to varying degrees. Ada offers one of the most complete implementations of polymorphism and generics. If you are new to generic programming you might want to do some exercises in Haskell or OCaml before attempting Ada generics.

  Make also sure you understand the differences between the frequently confused concepts of object orientation and generics. Object orientation (as in Java-, Python-, C#-style object orientation) is not listed as an essential concept here as it is mostly incompatible with real-time systems. (C++-, Ada-, Rust- or Swift-style object orientation qualifies for real-time systems under certain conditions.)
- **Contracts**, specifically **pre**- and **post**-conditions on operations and **invariants** on types.

In-depth knowledge of the above items are expected by any computer science student who studies the field as his or her main subject.

- **Tasking** (creation, termination).
- **Passive synchronization**: especially **Monitors** (**protected** procedures, entries and functions).
- **Message passing**: especially **synchronous message passing**.
- **Non-deterministic select-statements**.
- Groups of interfaces (‘**entry families**’).
- **Abstract** types and **dispatching.**

The last group of concepts is an essential pre-condition for this course (and is in fact expected from a third year computer science student). If you are struggling with those you have until week three to fill in. You also *need* to ask for help in this case. Form a small learning group or ask any of us.

Practical survival hints: work like a professional and use the accessible tools, i.e. e.g. set up your tool chain in the beginning of the course. Use the Gnat-Programming-Studio (GPS) instead of a plain text editor (just type `gps&` inside the directory where your (downloaded) project is and all will come up fine). Use your preferred version-management system. Make sure all your work is backed up ("the dog ate my memory stick" does not count in a fourth year course any more). The time with your tutor and myself is sparse and valuable. Make best use of it and come to all labs and lectures prepared and with good questions. If you need to catch up on the material above, you will need to spend a *significant* amount of time in preparations on your own or in a small learning group. Be as honest with yourself as you can be. If in doubt always re-check, -confirm your knowledge with somebody else. One of the most important functions of your tutor is to listen to your explanations and questions and to give you hints about how to study most productively. Yet the initiative and the questions have to come from you!