SOFTWARE ENGINEERING

RWTH AACHEN UNIVERSITY

http://www.se.rwth-aachen.de/

Department of Software Engineering
RWTH Aachen University
Prof. Dr.-Ing. Manfred Nagl, Emeritus,
Dipl.-Math. Michael von Wenckstern
(vonwenckstern@se-rwth.de)

Exercise course: *Ada*
WS 2014 / 15
October 22, 2014

Exercise Sheet 2

Submission:

When:	Thu, **October 28th**, 2014. 11:59 pm
Where:	L2P-eLearning room of Ada or e-mail

Organization

Exercise sheets must be submitted in groups of two to four students. The submission must be delivered electronically via the L2P-eLearning room of Ada.

Exercise 2.1 Control Flow Structures (3 points)

a) Why does the following sequence of statements evoke a compile error?

```
with Ada.Text_IO;

use Ada.Text_IO;

...

x : Integer;

...

case x is

      when 1 => Put_Line(„Eins");

      when 2 => Put_Line(„Zwei");

end case;
```

b) Modify the following loop, so that the outer loop is also stopped by the exit statement, but without replacing it.

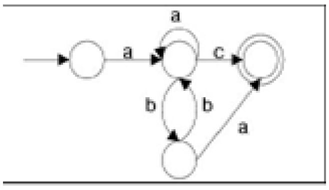
```
loop
        S1;
        loop
                S2;
                exit when B1;
        end loop;
end loop;
```

c) Write a for-loop, which calls the procedure `Zeichne` with actual parameter values 1.0, 1.2, 1.4, …, 10.0. `Zeichne` is declared as follows:

```
procedure Zeichne(x: in float);
```

Exercise 2.2 Programing Task (6 points)

Implement the following finite automata in Ada. For this, use the operation GET(c: CHARACTER) from the package Text_IO.



If a not allowed character is read in a current state, an appropriate error message should be given. If the final state is reached, then an OK-message should be given.

Exercise 2.3 Exception Handling (3 points)

a) What is the difference between a raise-statement with associated exception handler and a jump to a program part which includes the same exception handler?

b) A program for navigation of airplanes includes the following procedure declarations:

```
procedure Get_Position1(Position: out Position_Type);
procedure Get_Position2(Position: out Position_Type);
procedure Get_Position3(Position: out Position_Type);
```

All procedures are implemented to compute the position but are realized due to security reasons with different algorithms. Sketch a procedure `Compute_Position` for calculation of the position value. Thereby, `Get_Position2` is used as back-up for `Get_Position1` etc. If an exception occurs in `Get_Position1` or `Get_Position2` the appropriate back-up procedure is raised. If in

`Get_Position3` also an exception occurs then the exception `Navigation_Failure` is raised.

Exercise 2.4 Programming Task – Error Handling (6 points)

In the syntactic analysis phase of a compiler the source code of a program is checked for syntactical correctness with respect to a given grammar. For that purpose, the source code is first divided into a sequence of tokens (lexical analysis). Afterwards, this sequence of tokens is checked against the given grammar. A program which realizes this is called parser.

The simplest parsers are recursive descent parsers or also called top-down parsers. In these programs, there are in general for each nonterminal symbol of the grammar procedures which try to reproduce the production rule for the nonterminal symbol with the read terminal symbols.

This exercise sheet includes an Ada program that represents such a parser for simple expressions. The program can be downloaded from the L^{2}P-elearning room of Ada.

a) Implement the function `GetSym` which reads the next token.

A parser must be fault-tolerant. That means, if the source code or rather the sequence of tokens do not form a correct word in the grammar, the analysis should not stop at that place but continue at a later point (checkpoint). In the above mentioned Ada program the procedure `Test` fulfils this task. It reads tokens as long as a checkpoint is found. By calling the procedure the checkpoint information is notified via the two sets S1 (valid following tokens) and S2 (checkpoint for superior procedure).

b) Modify the parser program, so that exceptions and exception handlers are used for error handling. Discuss by this example the (dis-)advantages of error handling with exceptions. Which version is more suitable?