

Tentamen i  
Exam in

KOMPILATORTEKNIK  
COMPILER CONSTRUCTION

Kurskod/Course: D7011E / SMD163  
Datum/Date: 2008 - 08 - 27  
Skrivtid/Duration: 5 timmar

Totalt antal uppgifter/  
Number of assignments: 8  
Totalt antal poäng/  
Max score: 40  
Godkäntgräns/  
Required score: 23

Jourhavande lärare/  
Responsible teacher: Viktor Leijon  
Telefon/Phone: 070-660 14 04

Tillåtna hjälpmedel/  
Permitted material: English Dictionary

---

Full, clearly expressed, answers are required for full credit.  
Explain intermediate steps where applicable. Your job here is to  
show me that you understand the subject.

You may give your answers in either Swedish or English.

1. (5 marks)
- What do we mean with the semantics of a programming language? Please explain both static and dynamic semantics.
  - Describe the input to the type checker.
  - Describe the output from the type checker.

2. (5 marks)
- For the regular expression  $aa(b^*)(a?)$  give both a non-deterministic finite automata and a deterministic finite automata which accepts the language of the regular expression.
  - Please outline the respective advantages of DFAs and NFAs.

3. (4 marks)
- The following questions require **only** true/false answers. Each correct answer gives 0.5 marks, each incorrect answer subtracts 0.5 marks.

- The set of languages that can be expressed by LL(1) grammars is a strict superset of that for LL(0).
- The regular expression that matches a string is unique.
- No parsing algorithm is sufficient for all languages.
- The output of a compiler is always assembly code.
- The following two regular expressions describe the same language:  
 $a+a$  and  $a^*a$ .
- Memory fragmentation is only a problem with garbage collected languages.
- The following grammar (where  $c, d$  and  $e$  are terminals) is left recursive:  
 $A \rightarrow Bc$   
 $B \rightarrow Bb \mid e$
- When constructing a top-down parser, the FIRST set can never be empty for a meaningful non-terminal.

4. (5 marks)

Construct of an **recursive descent** parser for the following grammar:

$$\begin{aligned} \text{Exp} &\rightarrow \text{ID} \mid \text{Exp Op Exp} \\ \text{Op} &\rightarrow '+' \mid '*' \end{aligned}$$

The quoted strings are literals. It is okay to assume that there are functions available in the environment to get the current token and consume tokens.

5.

(6 marks)

Below follows an excerpt from a formal type system definition for a language .

$$\begin{array}{c}
 \frac{E(v) = t \quad E \vdash e : t}{E \vdash v = e \text{ well-typed}} \qquad \frac{E \vdash e : \text{bool} \quad E \vdash s \text{ well-typed}}{E \vdash \text{while } (e) \text{ s well-typed}} \\
 \\
 \frac{E(v) = t}{E \vdash v : t} \qquad \frac{E \vdash e : \text{int} \quad E \vdash e' : \text{int}}{E \vdash e + e' : \text{int}} \qquad E \vdash n : \text{int} \\
 \\
 \frac{E(f) = t(t_1 \dots t_n) \quad E \vdash e_1 : t_1 \quad \dots \quad E \vdash e_n : t_n}{E \vdash f(e_1 \dots e_n) : t} \qquad \frac{E \vdash e : \text{int} \quad E \vdash e' : \text{int}}{E \vdash e < e' : \text{bool}} \\
 \\
 \frac{E \vdash e : \text{boolean} \quad E \vdash s_1 \quad E \vdash s_2}{E \vdash \text{if } (e) \text{ s}_1 \text{ else } s_2} \qquad \frac{E \vdash s_1 \quad \dots \quad E \vdash s_n \text{ all well-typed}}{E \vdash \{ s_1 ; \dots s_n ; \} \text{ well-typed}}
 \end{array}$$

Here the meta symbol  $e$  stands for an expression,  $s$  stands for a statement, while  $t$ ,  $v$ , and  $n$  range over types, variables, and integer constants respectively.

- a) Discuss informally which types are needed for the variables, in the following statement block:

```

{
a = 3; b = true;
if (a == 3) { c = b; d = d + a; e = (b == c); }
}

```

- b) Using the rules above, construct a formal deduction that the statement block in (a) is well-typed in an environment  $E$  that meets your criteria from a).

6.

(5 marks)

- a) What functionality does a Run-Time system provide?  
b) What is the principal difference between stack allocated and heap allocated memory?

7. (4 marks)

Define the term strength reduction and explain how it affects performance. Give an example of code before and after this optimisation is done.

8. (6 marks)

Please discuss what impact the following three potential changes would have on a MiniJava compiler. Describe which parts of the compiler you would change and how.

- a) Limiting variable names to a maximum length of three characters.
- b) Making all variables global, so that all occurrences of the variable `a` refer to the same variable.
- c) Forbidding infinite `while`-loops by introducing the rule that a program may never execute the body of a `while`-loop more than 100 times.