Mälardalen University TEN1 DVA230 2016
School of innovation, design and engineering

Compiler theory

Thursday, August 18, 14:10 - 18:30
Teacher: Daniel Hedin, 021-107052 (15:00 - 17:00)
Allowed aids: Any books, lecture notes and written notes

The exam consists of 44 points distributed over 13 questions. Answers must be given in English or Swedish and should be clearly justified.
Grades:

3	22p
4	32p
5	38p
Max	44p

- Explain all solutions. A correctly explained solution with minor mistakes may render full points.
- Write clearly. Unreadable solutions will not be graded.
- Start each question on a new page and only write on one side of the page.
- Write down any assumptions you make.
- NOTE: each question can contain more than one part that needs an answer. Read carefully, and make sure you answer everything!

Lexical analysis - regular expressions (5p)

1) Write a regular expression that can handle one line string literals with escapes. Examples of valid string literals are `"string"`, and `"\""`, while `"str"ing"` and `"\\""` are not valid string literals. (3p)

2) Is the following language regular or not? Justify your answer. (2p)

$$L ::= a \mid b \mid (\, L \,)$$

Grammars (8p)

3) What is an LL(1) grammar and why does it matter to recursive descent parser? (3p)

4) Is it possible to create a context-free grammar that ensures that a function is always called with the right number of arguments, i.e., where

```
void f(int a) { ... }

f(1);
f(2);
```

is syntactically correct, but where

```
void f(int a) { ... }

f(1);
f(1,2);
```

is not? Justify your answer. (2p)

5) Let X range over variables. Rewrite the following grammar to be unambiguous without modifying the generated language. (3p)

$$E \;::=\; \mathit{define}\; X \;\mathit{in}\; E \mid E, E \mid X$$

Derivation trees and abstract syntax (6p)

6) When is a grammar unambiguous? (2p)

7) Give the derivation tree and the abstract syntax tree for `3 + ( 4 + x )` given the following grammar.

$$E \;\rightarrow\; VAR \mid NUM \mid E + E \mid (\, E \,)$$

What is the difference between the two? (4p)

Parsing (8p)

8) Write pseudo code for a recursive descent parser for the following language. (6p)

$$\begin{array}{lcl} E & \rightarrow & T\,U \\ T & \rightarrow & T a \mid \lambda \\ U & \rightarrow & a \mid a\, b \end{array}$$

9) Can all context-free grammars be parsed with an SLR parser? Justify your answer. (2p)

Type Checking (8p) Consider a simple language of assignments and the corresponding type language

$$\begin{array}{lcl} s & \rightarrow & x = e \mid \tau\, x \mid s_1; s_2 \\ e & \rightarrow & n \mid b \mid x \\ \tau & \rightarrow & \mathit{int} \mid \mathit{bool} \end{array}$$

where x denotes variable names (identifiers), n denotes integers and b denotes booleans. Given the following pseudo code for a type system for the language

```
check(tenv, x = e):
  t1 = tenv[x]; t2 = check e; if (t1 != t2) error

check(tenv, s1 ; s2):
  check(tenv, s1); check(tenv, s2)

check(tenv, t x):
  if (x defined in tenv) error; tenv[x] = t

check(tenv, n): return int
check(tenv, b): return bool
check(tenv, x):
  if (x not defined in tenv) error;
  return tenv[x]
```

10) Consider the following extension with code blocks

$$\begin{array}{lcl} s & \rightarrow & x = e \mid \tau\, x \mid s_1; s_2 \mid \{s\} \\ e & \rightarrow & n \mid b \mid x \\ \tau & \rightarrow & \mathit{int} \mid \mathit{bool} \end{array}$$

Modify the pseudo code to handle this extension in such a way that, e.g.,

```
int x; { bool x; x = true }; x = 5
```

and

```
int x; { x = 5 }
```

are type correct, but

```
{ bool y }; y = true
```

is not. (6p)

11) Show that the following program is type correct. (2p)

```
int x; { bool x; x = true }; x = 5
```

Code generation. (9p)

12) Consider the following small language with integer functions with one argument.

$$\begin{array}{lll} s & \rightarrow & x = e \mid \mathit{int}\ x \mid s_1 ; s_2 \mid \mathit{function}\ f(x)\ \mathit{begin}\ s\ \mathit{end} \mid \mathit{return}\ e \\ e & \rightarrow & n \mid x \mid e + e \mid f(e) \end{array}$$

Write pseudo code for a code generator that takes a program in the above language and produces Trac-42 stack code. (6p)

13) Generate Trac-42 code for the following program. (3p)

```
int x;
function f(x) begin
  return x + x
end;
x = f(f(2))
```