Försättsblad till skriftlig tentamen vid Linköpings universitet

(fylls i av ansvarig)

Datum för tentamen	11-12-15
Sal	TER2
Tid	14.00-18.0
Kurskod	TDDB44
Provkod	TEN1
Kursnamn/benämning	Kompilatorkonstruktion
Institution	*IDA*
Antal uppgifter som ingår i tentamen	12
Antal sidor på tentamen (inkl. försättsbladet)	5 blad
Jour/Kursansvarig	Jour: Kristian Stavåker
Telefon under skrivtid	0763-361742, 013-284093
Besöker salen ca kl.	ca. 15.00 och 16.30
Kursadministratör (namn + tfnnr + mailadress)	Gunilla Mellheden, 013-282297 gunme@ida.liu.se
Tillåtna hjälpmedel	(Se tentamen)
Övrigt (exempel när resultat kan ses på webben, betygsgränser, visning, övriga salar tentan går i m.m.)	
Vilken typ av papper ska användas, rutigt eller linjerat	Rutat
Antal exemplar i påsen	62

Tentamen/Exam
TDDD16 Kompilatorer och interpretatorer / Compilers and Interpreters
TDDB44 Kompilatorkonstruktion / Compiler Construction
TDDD55 Kompilatorer och interpretatorer / Compilers and Inetrpreters

2011–12–15, 14.00 – 18.00

Hjälpmedel / Allowed material:

- Engelsk ordbok / Dictionary from/to English to/from your native language
- Miniräknare / Pocket calculator

General instructions:

- Read all assignments carefully and completely before you begin
- **Note that not every problem is for all courses.** Watch out for comments like "TDDD16 only".
- You may answer in Swedish or in English.
- Write clearly — unreadable text will be ignored. Be precise in your statements — unprecise formulations may lead to reduction of points. Motivate clearly all statements and reasoning. Explain calculations and solution procedures.
- The assignments are *not* ordered according to difficulty.
- The exam is designed for 40 points (per course). You may thus plan 6 minutes per point.
- Grading: U, 3, 4, 5 resp. Fx, C, B, A.
- The preliminary threshold for passing (grade 3/C) is 20 points.

1. (TDDD16 and TDDD55 only - 6p) **Formal Languages and Automata Theory**
Consider the language L consisting of all strings w over the alphabet $\{a, b\}$ such that if w contains an even number of a's then w must end in b. E.g. *ababa*, *aabaab*, *bbbbab* belong to L.

   (a) (1.5p) Construct a regular expression for L.

   (b) (1.5p) Construct from the regular expression an NFA recognizing L.

   (c) (2.5p) Construct a DFA recognizing L, either by deriving it from the NFA or by constructing it directly.

   (d) (0.5p) Give an example of a formal language that is not context-free.

2. (3p) **Compiler Structure and Generators**

   (a) (2p) What are the generated compiler phases and what are the corresponding formalisms (mention at least 5) when using a compiler generator to generate a compiler?

   (b) (1p) Most modern compilers have not just one but several intermediate representations. Explain how these are, in general, related to each other, and what this organization means for the code generation process.

3. (2p) **Symbol Table Management**
Describe what the compiler — using a symbol table implemented as a hash table with chaining and block scoped control — does in compiling a statically scoped, block structured language when it handles:

   (a) (0.5p) block entry

   (b) (0.5p) block exit

   (c) (0.5p) a variable declaration

   (d) (0.5p) a variable use.

4. (6p) **Top-Down Parsing**
Given a grammar with nonterminals S, P, Q, and R, where S is the start symbol, and the following productions:

```
1. S ::= P $ P
2. P ::= P £ Q
3.     | Q
4. Q ::= R # Q
5.     | R
6. R ::= 0
7.     | 1
```

Assume that $, £, and # are operators.

(a) (1p) What is the associativity (right, left, none) of the operators?

(b) (1p) What is the precedence (relative priority) between the operators?

(c) (2p) Can the grammar be used directly for a recursive-descent parser? Motivate your answer. If not, rewrite the grammar so that the language it defines can be parsed using the recursiuve-descent method.

(d) (2p) Write a recursive-descent parser to analyze the language defined by the grammar.

5. (TDDB44 and TDDD16 only - 6p) **LR parsing**
Given the following grammar G for strings over the alphabet {a,b,p,q} with nonterminals A and B, where A is the start symbol:

```
A ::= aAa | bAb | aBb | bBb | p
B ::= aBa | bBb | aAb | bAa | q
```

Is the grammar SLR(1)? Is it LR(0)? Justify your answer using the LR item sets.
Construct the characteristic LR item NFA, the corresponding GOTO graph, and the ACTION and GOTO tables.
Show, using tables and stack, how the string aabqbba is parsed.

6. (TDDD55 only - 6p) **LR parsing**

(a) (3p) Use the SLR(1) tables below to show how the string a&b#a%b is parsed. You should show, step by step, how stack, input data etc. are changed during the parsing. Start state is0, start symbol is S.

Grammar:

```
1. S ::= X # X
2. X ::= Y % X
3.     | Y
4. Y ::= Y & Z
5.     | Z
6. Z ::= a
7.     | b
```

Tables:

	Action									
State	$	#	%	&	a	b	S	X	Y	Z
00	*	*	*	*	S09	S10	1	2	5	8
01	A	*	*	*	*	*	*	*	*	*
02	*	S03	*	*	*	*	*	*	*	*
03	*	*	*	*	S09	S10	*	4	5	8
04	R1	*	*	*	*	*	*	*	*	*
05	R3	R3	S06	S11	*	*	*	*	*	*
06	*	*	*	*	S09	S10	*	7	5	8
07	R2	R2	*	*	*	*	*	*	*	*
08	R5	R5	R5	R5	*	*	*	*	*	*
09	R6	R6	R6	R6	*	*	*	*	*	*
10	R7	R7	R7	R7	*	*	*	*	*	*
11	*	*	*	*	S09	S10	*	*	*	12
12	R4	R4	R4	R4	*	*	*	*	*	*

(b) (3p) Explain the concept of conflict in LR parsing — what it is, how it could be handled.

7. (3p) **Error Handling**
   Explain, define, and give examples of using the following concepts regarding error handling:

   (a) (1p) Valid prefix property,

   (b) (1p) Phrase level recovery,

   (c) (1p) Global correction.

8. (5p) **Syntax-Directed Translation**
   A Pascal-like language is extended with a `restartblock` statement according to the following grammar:

```
<block>     ::= begin <stmt_list> end
<stmt_list> ::= <stmt_list><stmt> |
<stmt>      ::= <assignment> | ... | restartblock
```

   (where "..." represents all other possible kinds of statements). `restartblock` means that execution restarts at the beginning of the immediately enclosing block.

   Example:

```
begin
    x:=17;
L1: begin
        y:=y-42;
        if p=4711
L2:        then restartblock
           else q:=q-1
L3: end;
end;
```

   where `restartblock` at L2 means a jump to L1 (i.e. the beginning of the enclosing block).

   (a) (4p) Write a syntax-directed translation scheme, with attributes and semantic rules, for the above grammar section.

   (b) (1p) What problem would occur in handling of the translation scheme if instead of `restartblock` there would be an `exitblock` statement that jumped to the end of the immidiately enclosing block (instead of the `begin`), i.e. to L3 in this example?

9. (3p) **Memory management**
   What does an activation record contain? What happens on the stack at function call? What happens on the stack at function return?

10. (6p) **Intermediate Code Generation**
Given the following code segment in a Pascal-like language:

```
if x<y
   then while x>z
              x:=x-10
   else y:=factorial(x);
```

(a) (3p) Translate the code segment into an abtract syntax tree, quadruples, and postfix code.

(b) (3p) Divide the following code inte basic blocks, draw a control flow graph, and show as well as motivate the existing loop(s).

```
    goto L2
L1: x:=x+1
L2: x:=x+1
L3: x:=x+1
    if x=1 then goto L1
    if x=2 then goto L3
    if x=3 then goto L5
L4: x:=x+1
L5: x:=x+1
    if x=4 then goto L4
```

11. (TDDB44 only - 6p) **Code Generation for RISC etc.**

(a) (2p) Explain the main characteristics of CISC and RISC architectures, and their differences.

(b) (1p) Explain what register allocation and register assignment is (in the context of code generation), and what the difference is.

(c) (3p) Given the following medium-level intermediate representation of a program fragment:

```
1: a = 1.0
2: b = 1.0
3: c = 3.0
4: e = 2.0
5: goto 9
6: b = a + b
7: a = c / 2.0
8: c = a * e
9: e = e / 2.0
10: f = (e > 0.1)
11: if f goto 6
12: d = 1.5 * a
```

Identify the live ranges of program variables, and draw the live range interference graph for the entire fragment. Assign registers to all live ranges by coloring the live range interference graph. How many registers do you need at least, and why?

12. (TDDB44 only - 3p) **Compiler Lab Exercises**
Correct and complete labs from the 2011 TDDB44 lab course handed in at the latest December 15, 2011, will give 3 points. State if you think that you have fulfilled the conditions and you should receive these points.