University of Cape Town

Department of Computer Science

CSC3003S Final Exam

2007

Marks : 35

Time : 3 hours

Instructions:

- Answer all questions from Section A and 3 questions from Section B.
- Show all calculations where applicable.

Section A [Answer Question ONE – this is compulsory]

Question 1 [8 marks]

1) What is the purpose of each of the following stages in a hypothetical compiler? [4]

   a) IR code generation

   convert abstract/concrete syntax tree to intermediate representation tree [1]

   b) Parsing

   derive grammatical/syntactical structure of program [1]

   c) Lexical analysis

   break input stream into tokens [1]

   d) Maximal Munch

   tile IR tree to select machine instructions [1]

2) Modern compilers are often divided into a front-end and back-end.

   a) Which of the 4 stages above are front-end activities and which are back-end? [2]

   front-end: lexical analysis [1/2], parsing [1/2]; back-end: code generation [1/2], maximal munch [1/2]

   b) Discuss 2 advantages of separating the front-end from the back-end. [2]

   easier to retarget compiler to new machine [1], easier to apply optimisations to IR [1], easier to build compiler for new language [1]

Section B [Answer 3 questions ONLY]

Consider the grammar and the LR(1) automaton for this grammar in Figures 1 and 2 below:

```
1.   S'   ->   S #
2.     S   ->   E
3.     E   ->   E - T
4.     E   ->   T
5.     T   ->   n
6.     T   ->   ( E )
```

Figure 1: A grammar for differences of numbers

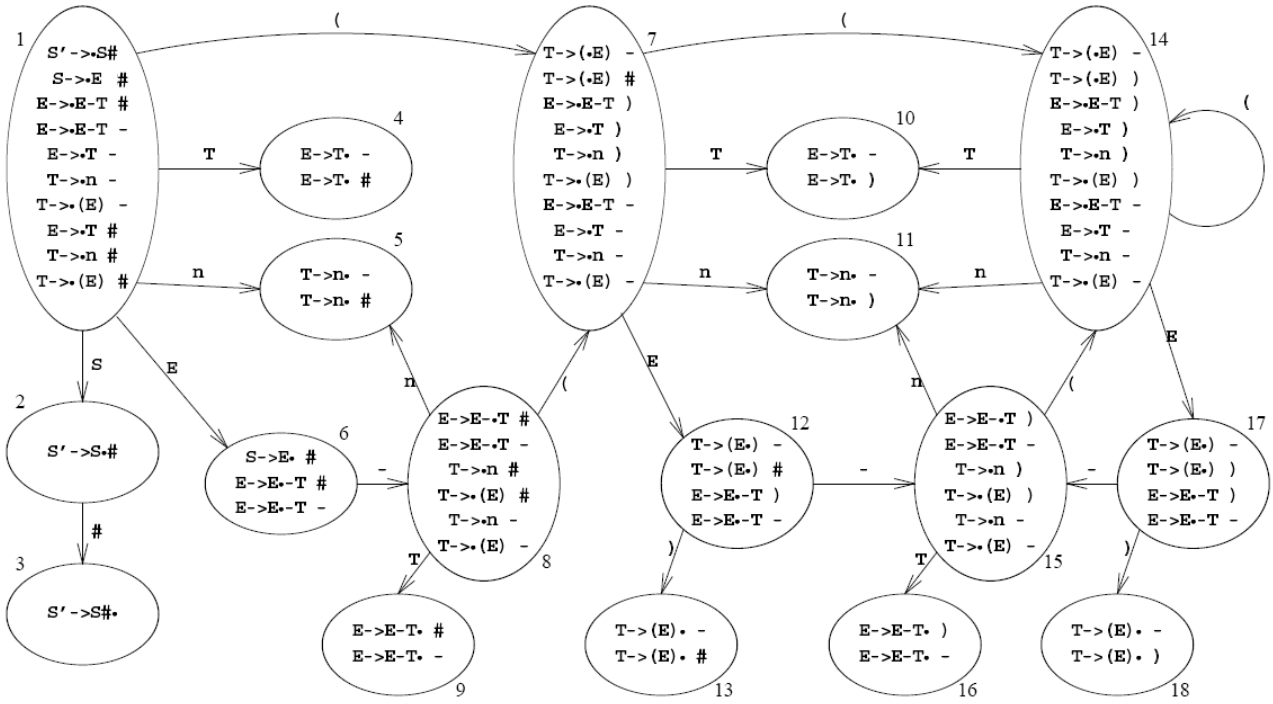

Figure 2: Deterministic LR(1) automaton for the grammar in Figure 1

Question 2: LR(1) Parsing [9 marks]

1) Complete rows 13 and 14 of the LR(1) parsing table in Figure 3 below:

	n	-	(	)	#	S	E	T
1	s5	e	s7	e	e	s2	s6	s4
2	e	e	e	e	s3			
3/acc								
4	e	r4	e	e	r4			
5	e	r5	e	e	r5			
6	e	s8	e	e	r2			
7	s11	e	s14	e	e		s12	s10
8	s5	e	s7	e	e			s9
9	e	r3	e	e	r3			
10	e	r4	e	r4	e			
11	e	r5	e	r5	e			
12	e	s15	e	s13	e			
13								
14								
15	s11	e	s14	e	e			s16
16	e	r3	e	r3	e			
17	e	s15	e	s18	e			
18	e	r6	e	r6	e			

Figure 3: LR(1) table for the grammar in Figure 1

Use the template below for your answer:

	n	-	(	)	#	S	E	T
13								
14								

Answer:

	n	-	(	)	#	S	E	T
13	e	r6	e	e	r6			
14	s11	e	s14	e	e		s17	s10

6 marks - 1 mark for each entry above, excluding error entries

2) Use the completed LR(1) parsing table from the previous question to parse the string n-n-n. Show only the first 3 steps of the parsing process.

Answer:

a	①	n-n-n#	shift
b	① n ⑤	-n-n#	reduce 5
c	① T ④	-n-n#	reduce 4

3 marks – 1 mark per step

Question 3: LALR(1) and SLR(1) Parsing [9 marks]

1) Complete state 2 of the LALR(1) automaton in Figure 4 below:

   E->T• [#-)]

   1 mark

2) Complete state 6 of the LALR(1) automaton in Figure 4 below:

```
T->(•E) [#-)]
E->•E-T [#-)]
 E->•T [#-)]
 T->•n [#-)]
T->•(E) [#-)]
```

   5 marks – 1 mark per element

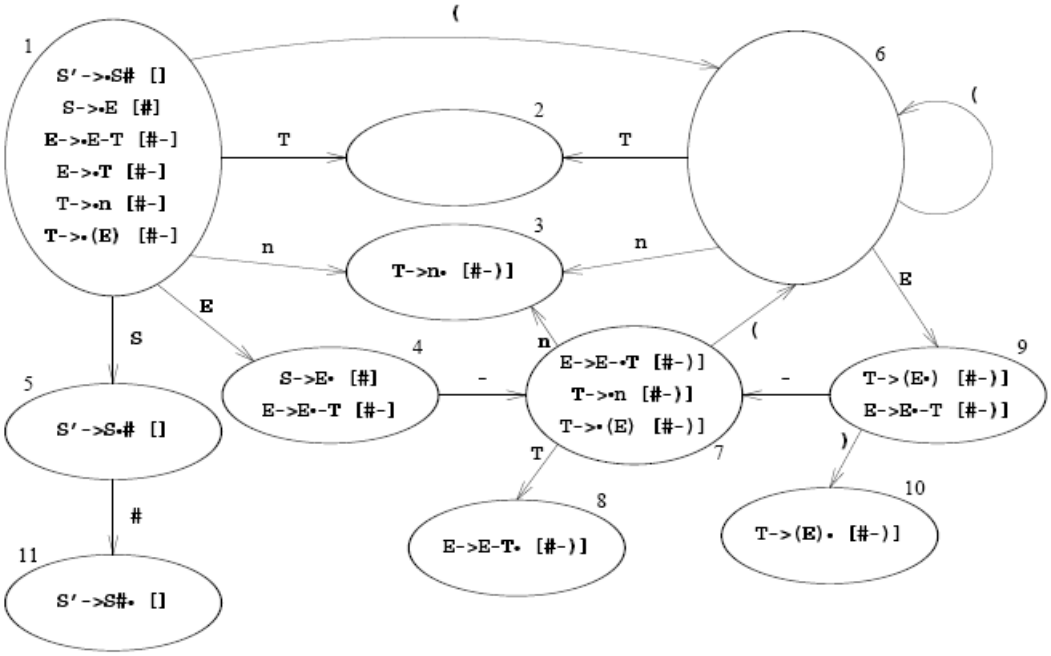

Figure 4: LALR(1) table for the grammar in Figure 1

3) Complete the missing look-ahead in state 1 of the LALR(1) automaton in Figure 5 below. Motivate your answer.

   Answer: [#] *1 mark*

4) Complete the missing look-ahead in state 2 of the LALR(1) automaton in Figure 5 below. Motivate your answer.

   Answer: [#-)] *1 mark*

5) Complete the missing look-ahead in state 3 of the LALR(1) automaton in Figure 5 below. Motivate your answer.

   Answer: [#-)] *1 mark*

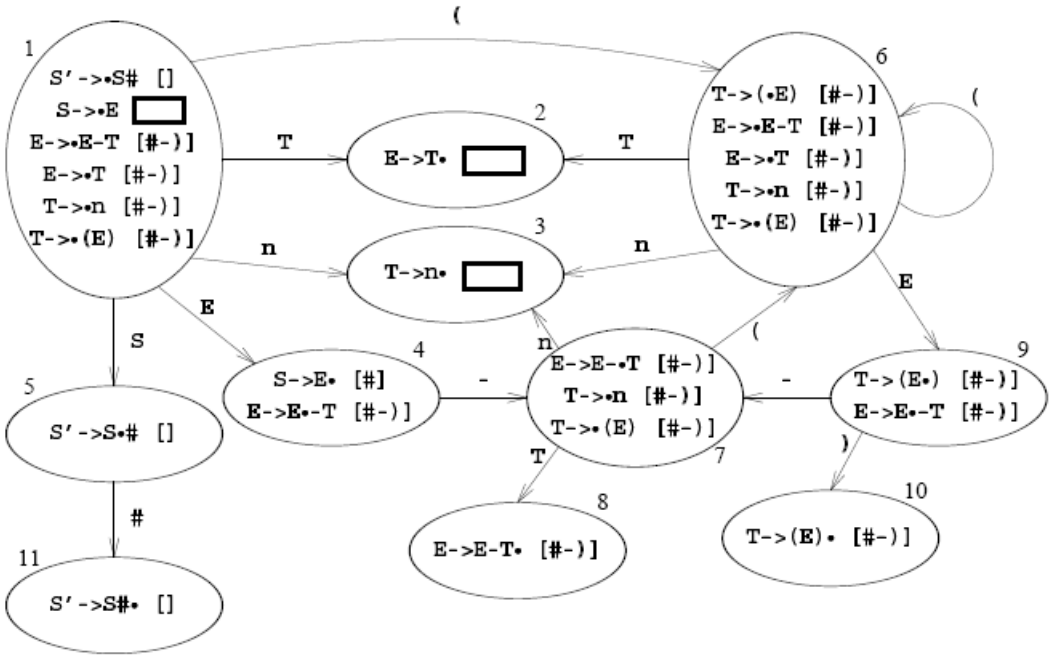

Figure 5: SLR(1) table for the grammar in Figure 1

Question 4: Code Analysis [9 marks]

a) Scope is a key concept in modern programming languages and compilers need to cater for this.

   a. Write a short method (in C, C++ or Java) that attempts to access an out-of-scope variable. Assume there are no global or instance variables available to this method. [1]

void test () { a = 1; }

   b. What mechanism is used by a compiler to detect such out-of-scope variables? [1]

symbol tables

   c. Briefly discuss 2 other context-sensitive errors that can be detected with the same mechanism. [2]

array index out of bounds [1], expression not type consistent [1], memory not allocated [1], …

b) When the program is deemed error-free, activation records are created for each subprogram.

   a. What is the purpose of the static link in an activation record? [1]

points to activation record of statically nested parent

   b. What is the purpose of the dynamic link in an activation record? [1]

points to caller's activation record

   c. What are the other 4 fields that can appear in a conceptual activation record? [2]

parameters [1], local vars [1], return address [1], return value [1]

   d. A display can be used for the same purpose as the static link. What is one advantage of using a display? [1]

no backing up of pointers on return [1]

Question 5: Code Generation [9 marks]

a) A modern compiler such as GCC allows programmers to specify how much inlining the compiler should apply.

   a. Explain with an example what inlining is. [2]

replace call to subprogram with subprogram body

sub a (int b) { x = b } a(5); → *x = 5*

   b. Inlining can be considered to be a peephole technique. Explain what a peephole optimisation technique is. [1]

applies to localised bit of code only

   c. Briefly discuss 2 other optimisations that may be applied to IR trees. [2]

constant folding [1], constant propagation [1], common subexpression elimination [1],…

b) After optimisations are applied, instructions can be selected using a tiling algorithm.

   a. Describe the steps of an algorithm to select instructions by tiling. [3]

start at root; find largest matching tile and cover nodes [1]; repeat for subtrees until whole tree is covered [1]; generate instructions in reverse order [1]

   b. This algorithm may be optimal – what does optimal mean in this context? [1]

no two tiles can be combined to form one with a lower cost [1]