Course Name: Programming Languages Translations
Course Code: CC473
Semester: Spring 2024

Assigned: 16/05/2024
Due: 20/05/2024

Sheet Six

Question One

Given the $LL(1)$ parse table for a grammar G (E is the start symbol)

	v	+	*	(	)	\$
E	$E\to TR$			$E\to TR$		
R		$R\to +TR$			$R\to\in$	$R\to\in$
T	$T\to FQ$			$T\to FQ$		
Q		$Q\to\in$	$Q\to *FQ$		$Q\to\in$	$Q\to\in$
F	$F\to v$			$F\to (E)$		

Trace the parsing steps of the sentences:

a) $v * (v + v) + v \$$
b) $v * (v + v) + v *(v + v)$

(Give the stack contents, remaining input at each step).

Question Two

Given the following LL(1) parsing table. Write down the details of parsing steps of the sentence: "x and x or x$", showing contents of stack and input buffer at each step. (E is the start symbol of the grammar).

	x	or	and	(	)	$
E	TQ			TQ		
Q		orTQ			∈	∈
T	FR			FR		
R		∈	andFR		∈	∈
F	x			(E)		

Question Three

Given the following LL(1) parsing table. Write down the details of parsing steps of the sentence: "((((x)(x))(x)(x))", showing contents of stack and input buffer at each step. (N is the start symbol of the grammar).

	x	(	)	$
N	UQ	UQ		
Q			∈	∈
U	FR	FR		
R			∈	∈
F	x	(N)		

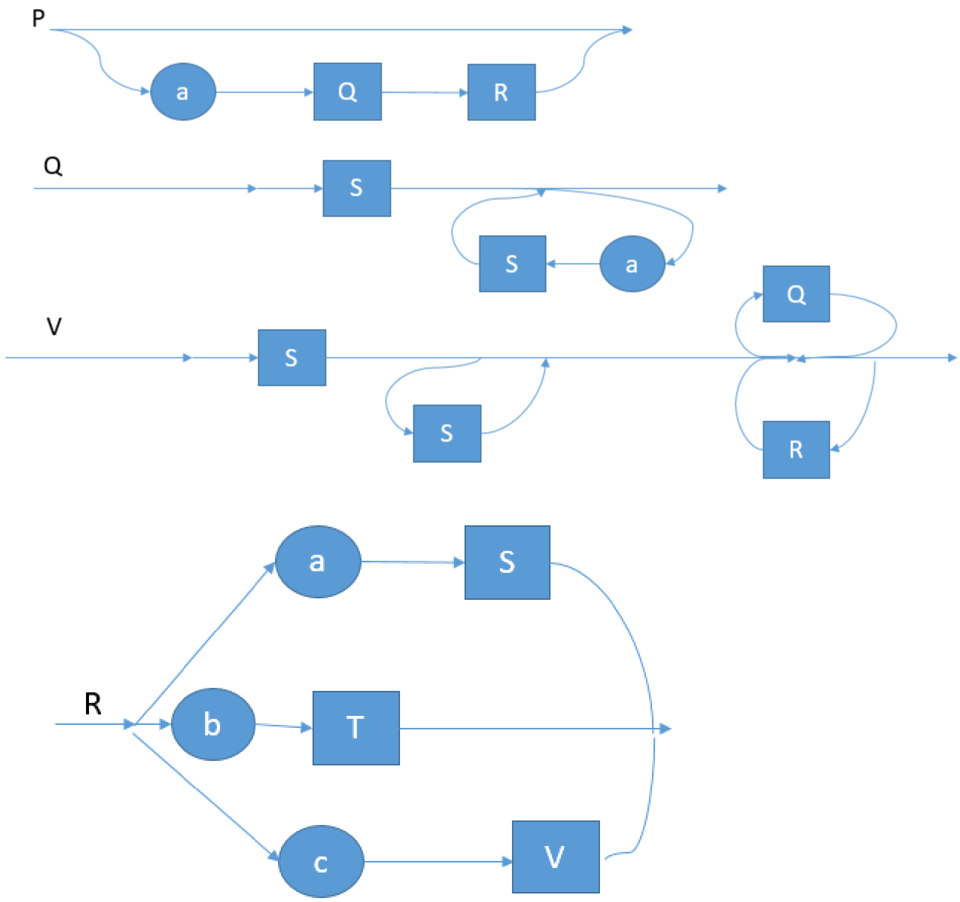
Question Four

Given the following LL(1) parsing table. Write down the details of parsing steps of the sentence: "(a#a)$", showing contents of stack and input buffer at each step. (X is the start symbol of the grammar).

	a	(	)	#	$
X	$X \to a$	$X \to (L)$			
L	$L \to XM$	$L \to XM$			
M			$M \to \mathcal{E}$	$M \to \#XM$	

Question Five

Write a recursive descent parser for the mini-language: (**use any pseudo-code**)



Question Six

a) Briefly describe the main purpose of contextual analysis.
b) Assuming statically typed programming languages, what are the kinds of constraints that will be verified at compile time?
c) What are the tasks of contextual analyzer?

Question Seven

a) In the context of *Contextual Analysis*, what do we mean by a ***block***?
b) State the main types of program's block structure. Give some examples of languages supporting each structure. Sketch the program block structure for each type.
c) Compare each of the structures mentioned in (b) with respect to:
   I. Program form.
   II. Scope rules.
d) Describe a possible structure for the identification table for a language with ***nested-block*** structure assuming static scoping. What are the basic operations on such table?

e) Describe a possible structure for the identification table for a language with ***flat-block*** structure assuming static scoping. What are the basic operations on such table?

f) Describe a possible structure for the identification table for a language with ***monolithic-block*** structure assuming static scoping. What are the basic operations on such table?

Question Eight

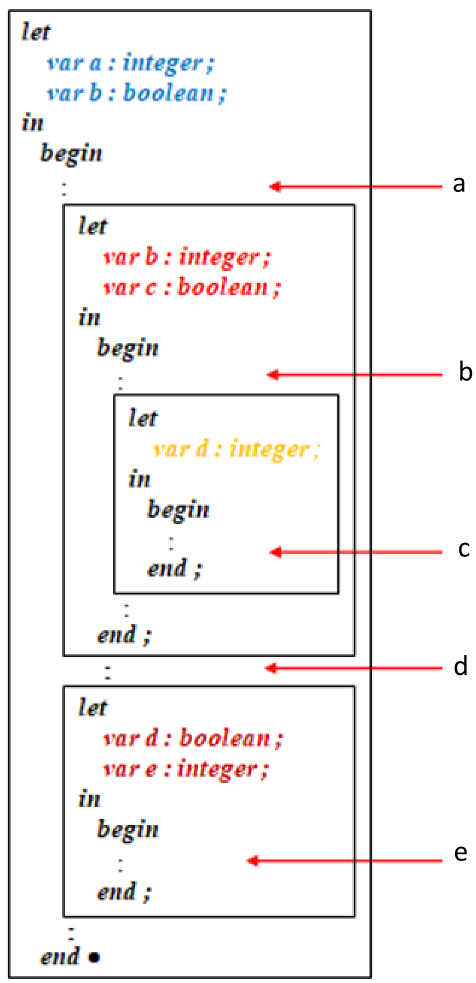
a) Describe in details, the term ***type checking***.

b) State the main definitions referred by ***type equivalence***.

c) Describe in details, how ***contextual analysis*** could be implemented.

Question Nine

Assuming static scoping, show the contents of the identification table of the following ***nested block structure*** program at each of the following breakpoints.



Question Ten

Consider the following expression:

[(A + B) * C + { (A + B) + E } * (E + F)] + [(A + B) * C]

a) Show the resultant Syntax Tree and DAG from the expression (remember that + and $*$ are commutative).
a) Show the resulting DAG when stored in three-address code.
b) Translate the expression to postfix form.
c) Translate the expression to indirect triple form.

Question Eleven

Consider the following expression:

$$((x * 4) + y) * (y + (4 * x)) + (z * (4 * x))$$

a) Show the resultant Syntax Tree and DAG from the expression (remember that + and $*$ are commutative).
b) Show the resulting DAG when stored in three-address code.
c) Show the resulting Syntax Tree when stored in three-address code
d) Translate the expression to postfix form.
e) Translate the expression to:
    i. quadruple,
    ii. triple,
    iii. indirect triple.

Question Twelve

Given the three-address code;

$$\begin{aligned}
1&: \ t1 = B * C \\
2&: \ t2 = A + t1 \\
3&: \ t3 = A * t1 \\
4&: \ t4 = t2 + t3 \\
5&: \ t5 = \sim C \\
6&: \ t6 = t4 / t5 \\
7&: \ D = t6
\end{aligned}$$

where "~" is the unary minus operator, and t1, t2, . . . , t6 are compiler-generated temporary names. Translate the expression to:

i. quadruple,
ii. triple,
iii. indirect triple.

Question Thirteen

Represent the following expression using three-address code as well as using acyclic directed graph

((X+Y) − ((X+Y) * (X-Y))) + ((X+Y) * (X-Y))

Question Fourteen

Consider the following expression:

$$[(A + B) * C + \{ (A + B) + E \} * (E + F)] + [(A + B) * C]$$

a) Show the resultant Syntax Tree and DAG from the expression (remember that + and ∗ are commutative).
b) Show the resulting DAG when stored in three-address code.
c) Translate the expression to postfix form.
d) Translate the expression to indirect triple form.

Question Fifteen

Consider the following expression:

$$((x * 4) + y) * (y + (4 * x)) + (z * (4 * x))$$

a) Show the resultant Syntax Tree and DAG from the expression (remember that + and ∗ are commutative).
b) Show the resulting DAG when stored in three-address code.
c) Show the resulting Syntax Tree when stored in three-address code
d) Translate the expression to postfix form.
e) Translate the expression to:
   i. quadruple,
   ii. triple,
   iii. indirect triple.

Question Sixteen

Represent the following assignment using DAG and three-address code:

$$a = b / (c + d) + b * (c + d) * (c + d)$$

Question Seventeen

Given the three-address code:

$$\begin{aligned} &1: \quad t1 = B * C \\ &2: \quad t2 = A + t1 \\ &3: \quad t3 = t1 * t2 \\ &4: \quad t4 = D + t3 \\ &5: \quad t5 = t2 + t4 \\ &6: \quad E = t5 \end{aligned}$$

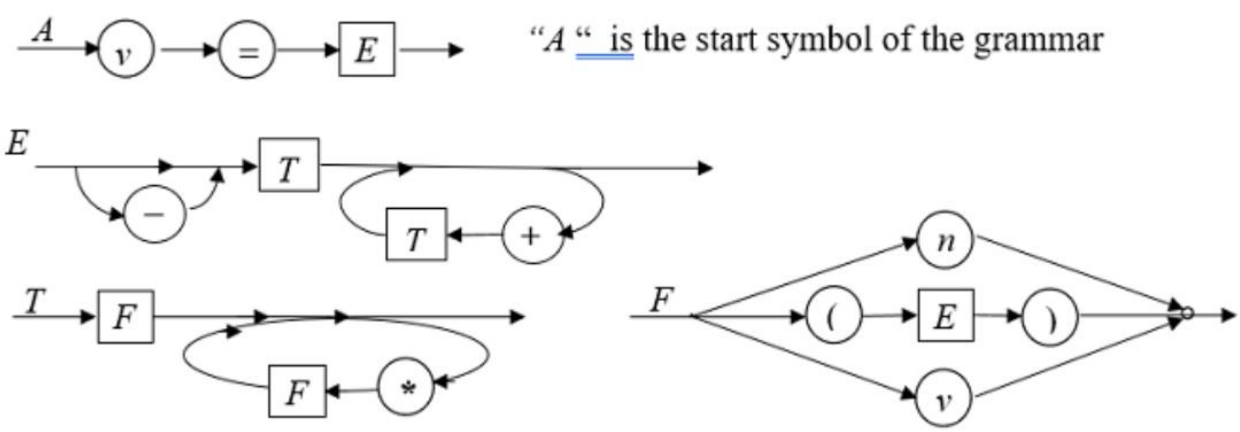
where t1, t2, . . . , t5 are compiler-generated temporary names. Translate this code to: *quadruples, triples*, and *indirect triples*.

Question Eighteen

A recursive-descent parser contains the following:
- A global variable, *LA* of type *token* to store the look-ahead symbol.
- A global procedure, *SCAN* to isolate next token (terminal symbol) from input buffer, store it in *LA* and finally remove it from input buffer.
- A procedure, *FAIL* to report unsuccessful parse and to stop run.
- A procedure, *CHECK* (*T*), with a parameter *T* of type token to recognize the expected terminal symbol *T*, defined as follows:

begin if *LA* = *T* ***then*** SCAN
else *FAIL*
end

Using these declarations, write down the definition for parsing procedures to process the non-terminals A , *E* , *T* , *F* and the driving procedure *main*. (Use any pseudo code).



Question Nineteen

State whether the following statements are true or false while correcting the incorrect (false) ones.

a) A bottom-up parser generates Left-most derivation.
b) Type checking is normally done during lexical analysis.
c) Symbol table is an intermediate representation of source program.
d) Relative to the program translated by a compiler, the same program when interpreted runs slower.
e) Lexical Analysis is specified by context-free grammar and implemented by pushdown automata.
f) Syntax Analysis is specified by regular grammar and implemented by finite-state machine.
g) Given the following expression grammar:
   $E \rightarrow E * F \qquad E \rightarrow F + E \qquad E \rightarrow F$
   $F \rightarrow F - F \qquad F \rightarrow id$
   i. * has higher precedence than +
   ii. − has higher precedence than *
   iii. + and − have the same precedence
   iv. + has higher precedence than *

Question Twenty

Compare and contrast the pros and cons of translating three address code into: quadruples, triples, and indirect triples.