Computer Science 330 Language Implementation Test Solution

Question 1 **15 Marks**

Write JFlex rules to match the following tokens, and where appropriate return a value, or perform some appropriate action.

(a) An octal integer. For example 077, 064, but not 0, 64, 069.

```
0[0-7]+ { return token( sym.OCTALINT ); }
```

(2 marks)

(b) A hexadecimal integer. For example 0xff, 0xFF, 0XFf, 0x123456789abcdef, but not ff, 0ff, xff.

```
0[xX][0-9a-fA-F]+{ return token( sym.HEXINT ); }
```

(2 marks)

(c) An identifier, possibly including underscores. For example, x1, hello, banana_boat, but not 1x.

```
[a-zA-Z_][a-zA-Z_0-9]* { return token( sym.IDENT ); }
```

(2 marks)

(d) A Windows, Macintosh or UNIX line break, with an action to increment a line count. Do not return a token.

```
 \r | \n | \r\n { lineCount++; }
```

(2 marks)

(e) Text enclosed in {:...:}. Add appropriate actions to increment the line count, when a line break occurs.

```
<NORMAL> {
      "{:" {yybegin( ACTION ); actionText = ""; }
      ...
      }
<ACTION> {
      ":}"  { yybegin( NORMAL ); return token( sym.ACTION, actionText ); }
      \r | \n | \r\n {
            lineCount++;  actionText += yytext(); }
      .     { actionText += yytext(); }
      }
```

(7 marks)

Question 2 65 marks

Consider the following CUP grammar.

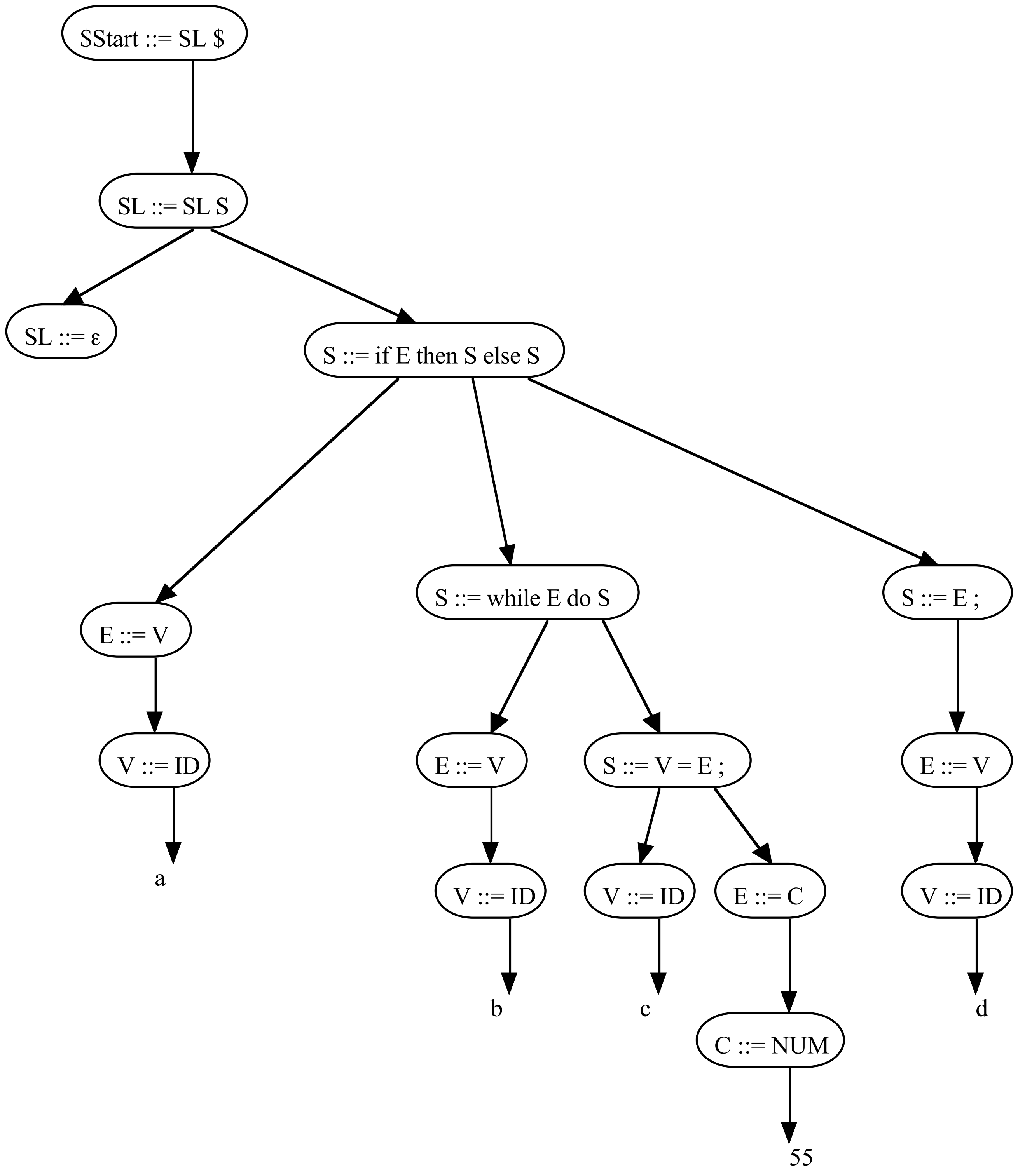
(a) Using the information provided in the appendix, perform a shift-reduce LALR(1) parse of the input

```
if a then while b do c = 55 ; else d ;
```

Show both the symbols and states on the stack, the current token, and the action performed at each stage.

Stack												Input	Action	
$0												if	Reduce	SL ::=
$0	SL 1												Shift	
$0	SL 1	if 10										a	Shift	
$0	SL 1	if 10	ID 4									then	Reduce	V ::= ID
$0	SL 1	if 10	V 16										Reduce	E ::= V
$0	SL 1	if 10	E 17										Shift	
$0	SL 1	if 10	E 17	then 18								while	Shift	
$0	SL 1	if 10	E 17	then 18	while 2							b	Shift	
$0	SL 1	if 10	E 17	then 18	while 2	ID 4						do	Reduce	V ::= ID
$0	SL 1	if 10	E 17	then 18	while 2	V 16							Reduce	E ::= V
$0	SL 1	if 10	E 17	then 18	while 2	E 28							Shift	
$0	SL 1	if 10	E 17	then 18	while 2	E 28	do 29					c	Shift	
$0	SL 1	if 10	E 17	then 18	while 2	E 28	do 29	ID 4				=	Reduce	V ::= ID
$0	SL 1	if 10	E 17	then 18	while 2	E 28	do 29	V 3					Shift	
$0	SL 1	if 10	E 17	then 18	while 2	E 28	do 29	V 3	= 25			55	Shift	
$0	SL 1	if 10	E 17	then 18	while 2	E 28	do 29	V 3	= 25	NUM 7		;	Reduce	C ::= NUM
$0	SL 1	if 10	E 17	then 18	while 2	E 28	do 29	V 3	= 25	C 8			Reduce	E ::= C
$0	SL 1	if 10	E 17	then 18	while 2	E 28	do 29	V 3	= 25	E 26			Shift	
$0	SL 1	if 10	E 17	then 18	while 2	E 28	do 29	V 3	= 25	E 26	; 27	else	Reduce	S ::= V = E ;
$0	SL 1	if 10	E 17	then 18	while 2	E 28	do 29	S 30					Reduce	S ::= while E do S
$0	SL 1	if 10	E 17	then 18	S 19								Shift	
$0	SL 1	if 10	E 17	then 18	S 19	else 20						d	Shift	
$0	SL 1	if 10	E 17	then 18	S 19	else 20	ID 4					;	Reduce	V ::= ID
$0	SL 1	if 10	E 17	then 18	S 19	else 20	V 3						Reduce	E ::= V
$0	SL 1	if 10	E 17	then 18	S 19	else 20	E 5						Shift	
$0	SL 1	if 10	E 17	then 18	S 19	else 20	E 5	; 24				$	Reduce	S ::= E ;
$0	SL 1	if 10	E 17	then 18	S 19	else 20	S 21						Reduce	S ::= if E then S else S
$0	SL 1	S 11											Reduce	SL ::= SL S
$0	SL 1												Shift	
$0	SL 1	$9										$	Reduce	$Start ::= SL $
$0	$Start -1												Accept	

(b) Draw the full parse tree, showing all rules used in the above shift-reduce LALR(1) parse.(8 marks)

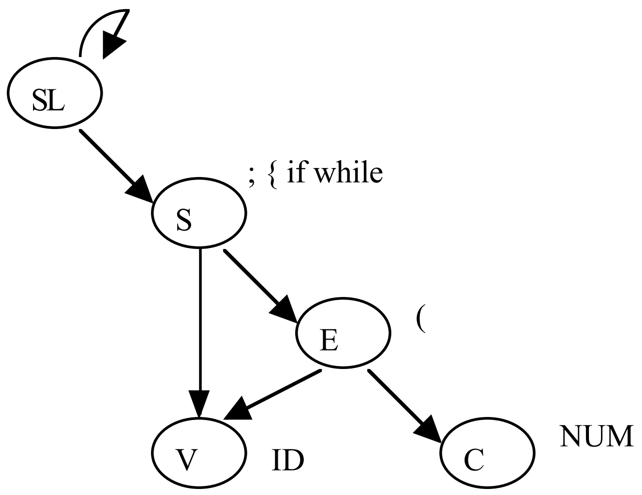


(c) (i) Note that StmtList is nullable.

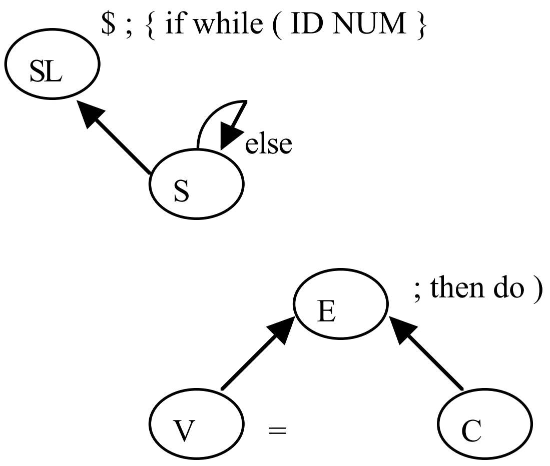
(ii) Draw the first graph, and compute the first sets for this grammar.

(iii) Draw the follow graph, and compute the follow sets for this grammar. (27 marks)

First Graph



Follow Graph



Symbol	First Set	Follow Set
SS	; { if while (ID NUM	$; { if while (ID NUM }
S	; { if while (ID NUM	$; { if while (ID NUM } else
E	(ID NUM	; then do)
V	ID	; then do) =
C	NUM	; then do)

(d) State 1 is ... Write down the set of items for goto(state 1, IF) (state 10). Make sure you take the closure.

```
lalr_state [10]: {
  [Stmt ::= IF (*) Expr THEN Stmt ,
      {EOF IF WHILE LEFT LEFTCURLY SEMICOLON NUMBER IDENT }]
  [Stmt ::= IF (*) Expr THEN Stmt ELSE Stmt ,
      {EOF IF WHILE LEFT LEFTCURLY SEMICOLON NUMBER IDENT }]
  [Expr ::= (*) LEFT Expr RIGHT , {THEN }]
  [Expr ::= (*) Constant , {THEN }]
  [Expr ::= (*) Variable , {THEN }]
  [Variable ::= (*) IDENT , {THEN }]
  [Constant ::= (*) NUMBER , {THEN }]
}
```

Question 3 20 marks

```
GrammarRule::= IDENT "::=" RHSList ";" ;
RHSList ::= RHS | RHSList "|" RHS ;
RHS::= SymbolList ActionOpt PrecOpt ;
SymbolList::= /* empty */ | SymbolList Symbol ;
Symbol::= IDENT | IDENT ":" IDENT ;
ActionOpt::= /* empty */ | ACTION ;
PrecOpt::= /* empty */ | PERCENTPREC IDENT ;
```