HW1

Graded

20 Hours, 34 Minutes Late

Student

Adam Fenjiro

Total Points

59 / 80 pts

Question 1

Q1 **8** / 10 pts

– **0 pts** Correct

– **2 pts** Incorrect rules and stack changes after step 3

Question 2

Q2 **21** / 30 pts

2.1 **a** **8** / 10 pts

– **0 pts** Correct

– **2 pts** Missing closure step in CFSM. The given grammar is SLR.

2.2 **b** **6** / 10 pts

– **0 pts** Correct

– **4 pts** Incomplete CFSM and missing first/follow sets

2.3 **c** **7** / 10 pts

– **0 pts** Correct

– **3 pts** Incomplete CFSM and parser table

Question 3

Q3 **30** / 40 pts

3.1 a **2** / 5 pts

– **0 pts** Correct

– **3 pts** Point adjustment

3.2 b **3** / 5 pts

– **0 pts** Correct

– **2 pts** Incomplete CFSM

3.3 c **4** / 5 pts

– **0 pts** Correct

– **1 pt** Point adjustment

3.4 d **2** / 5 pts

– **0 pts** Correct

– **3 pts** Point adjustment

3.5 e **10** / 10 pts

– **0 pts** Correct

3.6 f **8** / 8 pts

– **0 pts** Correct

3.7 g **1** / 2 pts

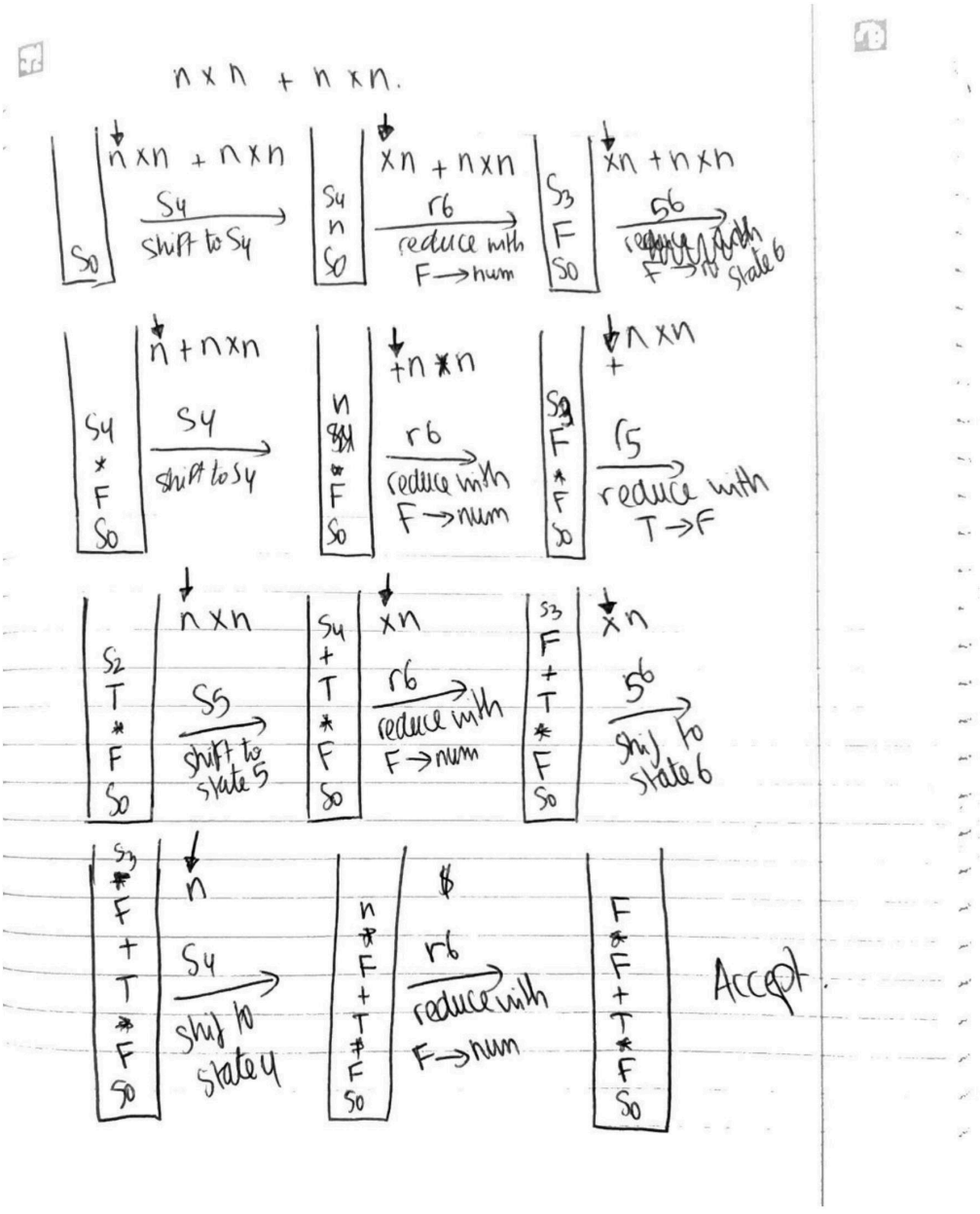
– **0 pts** Correct

– **1 pt** Point adjustment

Question assigned to the following page: 1

CS4121 - Spring25 - Adam Fenjiro - HW1

Problem 1:



Question assigned to the following page: 2.1

Problem 2:

a) S → abS
| ab

1. CFSM

State 0:
S' -> •S
S -> •abS
S -> •ab
State 1:
S' -> S•
State 2:
S -> a•bS
State 3:
S -> ab•S
State 4:
S -> abS•
State 5:
S -> ab•

2. SLR Parse Table

State	a	b	$	S
S0	s2	s5		1
S1			acc	
S2		s3		
S3	s2	s5		4
S4			r2	
S5			r3	

3. SLR or not

=> The grammar is not SLR because state 5 has a shift-reduce conflict. When the input is b the parser does not know whether to shift or reduce leading to ambiguity.

Question assigned to the following page: 2.2

b) S → AaAb
| BbBa
A → c
B → c

1. CFSM

State 0:
S' -> •S
S -> •AaAb
S -> •BbBa
A -> •c
B -> •c
State 1:
S' -> S•
State 2:
S -> A•aAb
A -> c•
State 3:
S -> B•bBa
B -> c•
State 4:
S -> Aa•Ab
S -> •AaAb
S -> •BbBa
A -> •c
B -> •c
State 5:
S -> Bb•Ba
S -> •AaAb
S -> •BbBa
A -> •c
B -> •c
State 5:
S -> AaAb•
S -> BbBa•

Question assigned to the following page: 2.2

2. SLR Parse Table

State	a	b	c	$	S	A	B
S0			S2		1	2	3
S1				acc			
S2	S4						
S3		S5					
S4			S2			6	7
S5			S3			8	9
S6				r1			

3. SLR or not

=> The grammar is not SLR. This is because there is a shift-reduce conflict in state 4 and 5. When the input is c the parser does not know whether to shift or reduce.

Question assigned to the following page: 2.3

c) S → ASB | ab
A → a
B → b

1. CFSM

State 0:
S' -> .•S
S -> •ASB
S -> •ab
A -> •a
B -> •b
State 1:
S' -> S•
State 2:
S -> A•SB
A -> a•
State 3:
S -> B•SB
B -> b•
State 4:
S -> AS•B
S -> •ASB
S -> •ab
A -> •a
B -> •b
State 5:
S -> ab•

2. SLR Parse Table

State	a	b	$	S	A	B
S0	S2	S3		1	2	3
S1			acc			
S2		S4				
S3	S5					
S4			r2			
S5			r3			

3. SLR or not

=> The grammar is SLR because there are no conflicts in the parse table. The parsing table correctly differentiates between shift and reduce actions, ensuring smooth parsing.

Questions assigned to the following page: 3.1 and 3.2

Problem 3:

a.

S' → E
E → +EE
E → + (+EE) E
E → + (+ (~E) E) E
E → + (+ (~5) 12) cy

b.

State 0:
S' -> •E
E -> •+EE
E -> •~E
E -> •F
F -> •num 10
F -> •num 26

State 1:
S' -> E•
E -> E•+E
E -> E•~E
E -> E•F

State 2:
E -> +•EE
E -> •+EE
E -> •~E
E -> •F
F -> •num 10
F -> •num 26

State 3:
E -> ~•E
E -> •+EE
E -> •~E
E -> •F
F -> •num 10
F -> •num 26

Questions assigned to the following page: 3.2, 3.3, and 3.4

State 4:
E -> F•
E -> F•+E
E -> F•~E
E -> F•F

State 5:
F -> num 10•

State 6:
F -> num 26•

c.

First Sets

First(E) = First(F) = { +, ~, num_{10}, num_{26} }

Follow Sets

Follow(S') = { $ }
Follow(E) = Follow(F) = { +, ~, num_{10}, num_{26}, $ }

d.

State	+	~	num10	num26	$	E	F
S0	S2	S3	S4	S4		1	5
S1					acc		
S2			S6	S6		7	5
S3	S3	S4	S4			8	5
S4							
S5					r4		
S6			S6	S6			
S7					r2		
S8					r3		

Question assigned to the following page: 3.5

e.

Flex Rules:

```
C/C++
%{
#include "y.tab.h"
%}

[0-9]+      { yylval.num = atoi(yytext); return NUM10; }
[a-z]+      { yylval.num = base26(yytext); return NUM26; }
"+"         { return '+'; }
"~"         { return '~'; }

%%
```

Bison Rules:

```
C/C++
%token NUM10 NUM26
%union { int num; }

%%
E : '+' E E { $$ = $2 + $3; }
  | '~' E   { $$ = -$2; }
  | F       { $$ = $1; }
F : NUM10   { $$ = $1; }
  | NUM26   { $$ = $1; }
%%
```

Base26 conversion in c

```
C/C++
int base26(char* str) {
    int result = 0;
    while (*str) {
        result = result * 26 + (*str - 'a');
        str++;
    }
    return result;
}
```

Questions assigned to the following page: 3.6 and 3.7

(f) Here is the parse process of the input ++~512cy:

Start by pushing S' into the stack

Read '+' first symbol from the input
Shift '+' then go to state 2

Read '+' next symbol from the input
Shift '+' and go to state 2

Read '~' next symbol from the input
Shift '~' and go to state 3

Read '5' next symbol from the input
Shift '5' and go to state 5

Reduce by rule F -> num 10 and go to state 4
Reduce by rule E -> F and go to state 1
Reduce by rule E -> ~E and go to state 2

Read 'c' next symbol from the input
Shift 'c' and go to state 6

Reduce by rule F -> num 26 and go to state 4
Reduce by rule E -> F go to state 1

Reduce by rule E -> +EE and go to state 2.
Reduce by rule E -> +EE and go to state 1.

Finally accept the input

g) They are both related since both of these find the same sequence of grammar rules to parse and derive the expression. The parsing process uses a stack and a parse table to systematically apply the grammar rules.