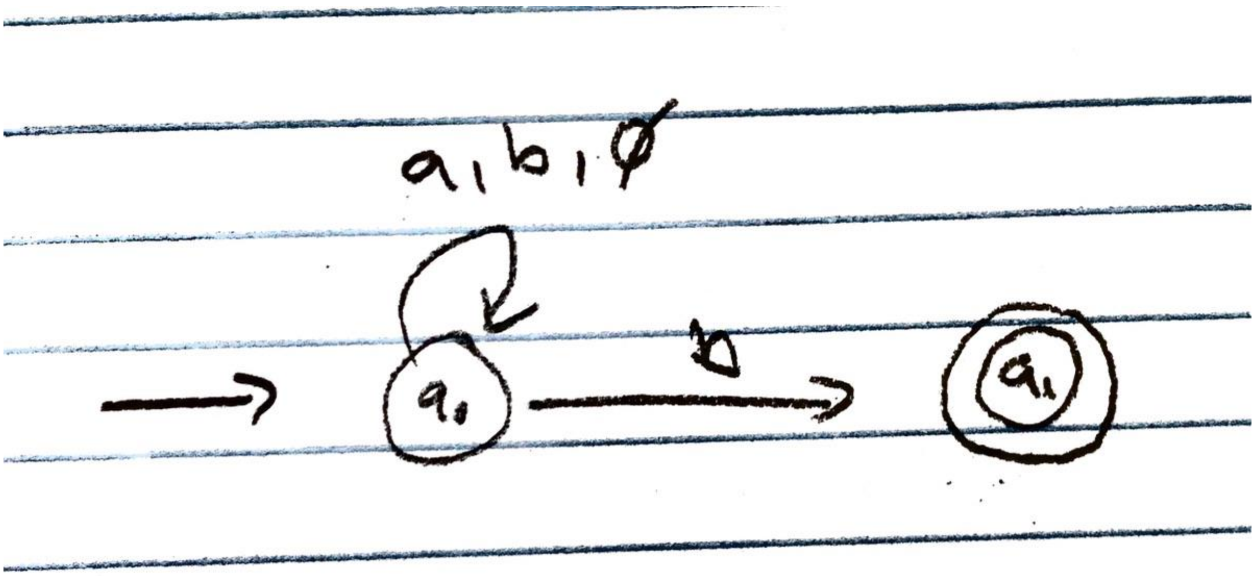
Homework 2: CFG

1. N.A.

2. . (15) Show the NFA that results from applying the NFA construction method, discussed in the class, to the regular expression (a | b | 0)∧ b.



3. (40) Context Free Grammar.

    a. Write a context free grammar for arithmetic expressions which can use numbers, variables and operation + only

        i. $expr \rightarrow expr + expr$

        ii. $expr \rightarrow var \mid num$

        iii. $var \rightarrow a \mid b \mid c$

        iv. $num \rightarrow 1 \mid 2 \mid \ldots \mid 9$

    b. Write a context free grammar for the arithmetic expressions above that captures right associativity.

        i. $expr \rightarrow var + expr \mid num + expr$

        ii. $var \rightarrow a \mid b \mid c$

        iii. $num \rightarrow 1 \mid 2 \mid \ldots \mid 9$

c. Write a context free grammar for arithmetic expressions which can use numbers, variables and binary operations - and + only. Your grammar has to capture the precedence that - must be computed before +. Your grammar should also capture the left associativity of + and -. For example 5+5−5−5+6−5 should be computed as 5 + (5 − 5 − 5) + (6 − 5).

   i. $expr \rightarrow operand \mid expr + group$

   ii. $var \rightarrow a \mid b \mid c$

   iii. $operand \rightarrow num \mid var$

   iv. $num \rightarrow 1 \mid 2 \mid \ldots \mid 9$

   v. $group \rightarrow expr - expr \mid expr$

d. Refine your grammar above to allow parenthesis (and). Hint. Recall the techniques on how precedence and associativity were dealt with in the class.

   i. $expr \rightarrow operand \mid expr + group \mid expr + (group)$

   ii. $var \rightarrow a \mid b \mid c$

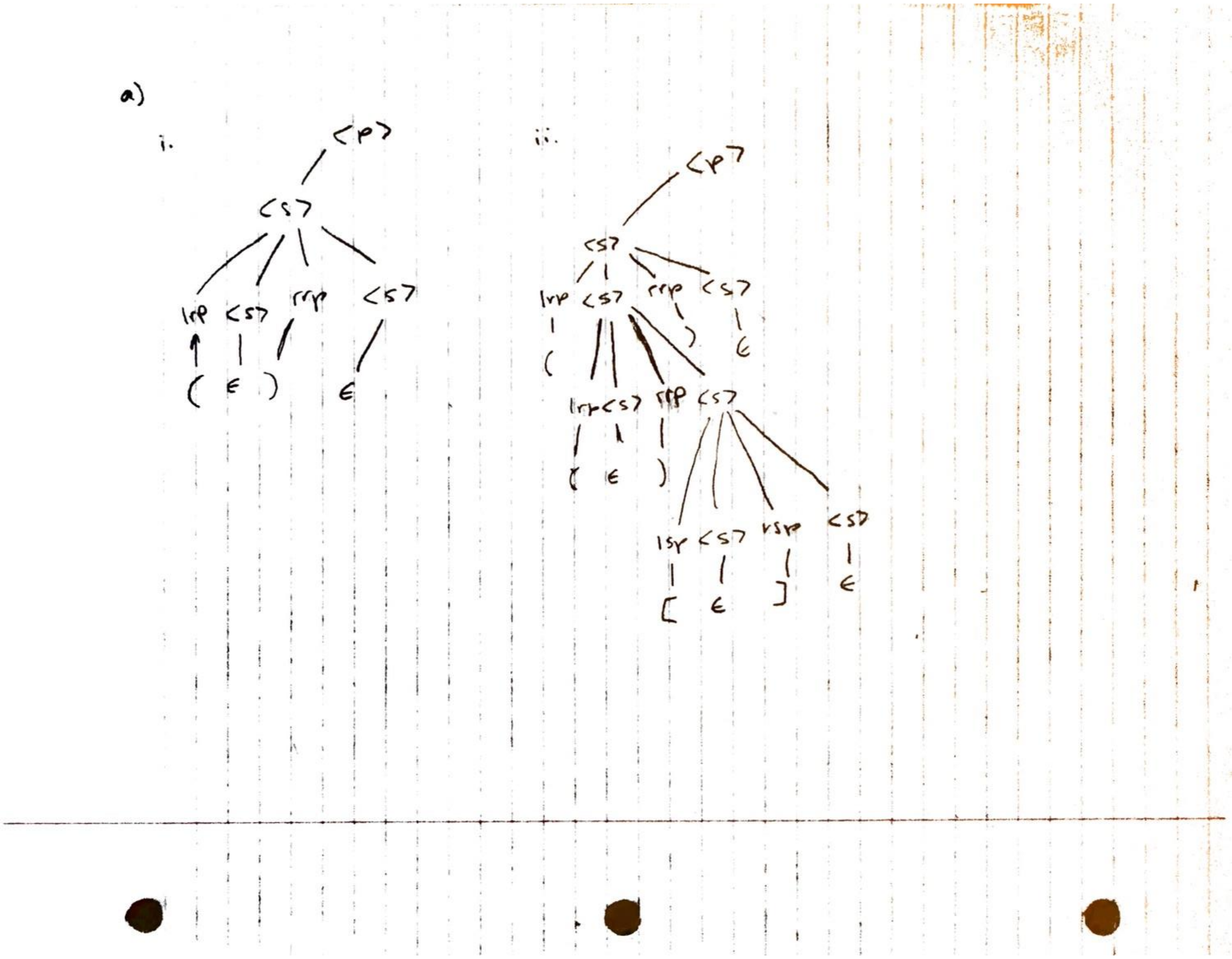
   iii. $operand \rightarrow num \mid var$

   iv. $num \rightarrow 1 \mid 2 \mid \ldots \mid 9$

   v. $group \rightarrow expr - expr \mid expr$

4. (15) Given a grammar < P >→< S > < S >→ lrp < S > rrp < S > | lsp < S > rsp < S > | ϵ The terminals are defined as follows lrp → (rrp →) lsp → [rsp →] CS3361 Concepts of Programming Languages by Y Zhang, TTU, Spring 2021

a. Draw a parse tree for each of the sentences:



b. Write a rightmost derivation for [].

i. $< P > \rightarrow < S >$

ii. $< S > \rightarrow lsp < S > rsp < S >$

1. $\rightarrow \; lsp < S > rsp \; \epsilon$

2. $\rightarrow \; lsp \; \epsilon \; rsp \; \epsilon$

3. $\rightarrow lsp \; rsp$

5. Consider the CFG for in the slides (of L6) for parsing (Page 16), and an input program sum := A + B

a)

Sum := A+B write sum

```
  <program>           Sum:= A + B write sum
  <stmt_list> $$             ↓
<stmt> <stmt_list> $$
 id := <expr> <stmt_list> $$   Match! continue..
 := <expr> <stmt_list> $$ | := A + B write sum; match!...
<expT> <stmt_list> $$ | A+B write sum;
<term> <term_tail> <stmt_list> $$ | A+B write sum;
<factr> <fact_tail> <term_tail> <stmt_list> $$ | A+B write sum;
 id <fact_tail> <term_tail> <stmt_list> $$ | match!
<fact_tail> <term_tail> <stmt_list> $$ | + B write sum;
  ε <term_tail> <statent_list> $$ | + B write sum;
<term_tail> <stmt_list> $$ | +B write sum;
 <add_op> <term> <term_tail> <stmt_list> $$ | + B write sum
  + <term> <term_tail> <stmt_list> $$ | match!
 <term> <term_tail> <stmt_list> $$ | B write sum
<factor> <fact_tail> <term_tail> <stmt_list> $$ | B write sum
 id <fact_tail> <term_tail> <stmt_list> $$ | match!
<fact_tail> <term_tail> <stmt_list> $$ | write sum
  ε <term_tail> <stmt_list> $$ | write sum
  ε <stmt_list> $$ | write sum
 <stmt> <stmt_list> $$ | write sum
  write <expr> <stmt_list> $$ | match!
  <expr> <stmt_list> $$ | sum
 <term> <term_tail> <stmt_list> $$ | sum
 <factr> <fact_tail> <term_tail> <stmt_list> $$ | sum
  id <fact_tail> <term_tail> <stmt_list> $$ | match!
    ε <term_tail> <stmt_list> $$ |
          ε <stmt_list> $$ |
              ε $$ | done
```