ธนภัทร์ เกิดเปี่ยม 65050368

Exercise 1

1. จาก Grammar

<expr> → <term> {(+ | -) <term>}

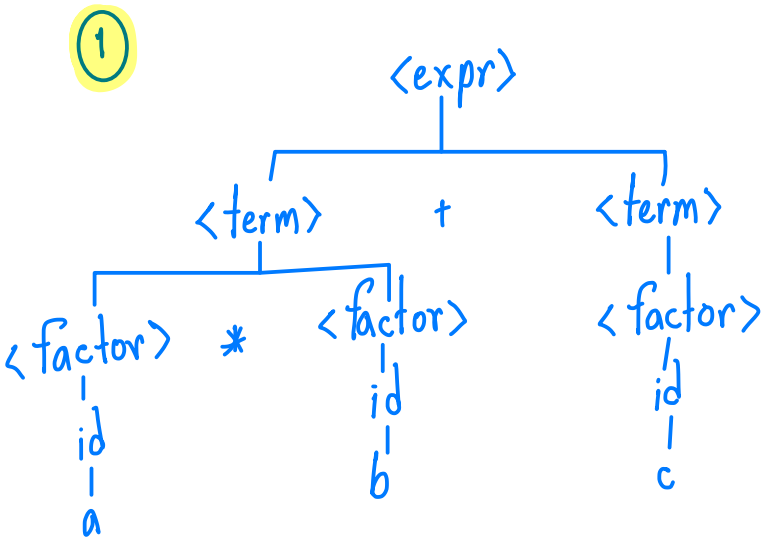
<term> → <factor> {(* | /) <factor>}

<factor> → id | (<expr>)

และฟังก์ชันต่อไปนี้

```
void expr() {
/* Parse the first term */
 term();
 ...
while (nextToken == PLUS_CODE ||
       nextToken == MINUS_CODE){
   lex();
   term();
 }
}
void term() {
          factor();
          while(nextToken == AST_CODE ||
         nextToken == SLASH_CODE)  {
                                lex();
                                factor();
  }
}
void factor() {
 /* Determine which RHS */
  if (nextToken) == ID_CODE)
 /* For the RHS id, just call lex */
    lex();
else if (nextToken == LEFT_PAREN_CODE) {
    lex();
    expr();
    if (nextToken == RIGHT_PAREN_CODE)
      lex();
    else
      error();
  } /* End of else if (nextToken == ... */
  else error(); /* Neither RHS matches */
 }
```

จงแสดงวิธีหา Derivation Tree ของสตริง a * b + c



call lex() // return a
Enter expr
Enter term
Enter factor
call lex() return *
Exit factor
call lex() return b
Enter factor
Enter term
call lex() return c
Enter term
Enter factor
call lex() // return end of file
Exit factor
Exit term
Exit expr

2. จาก Grammar ต่อไปนี้

1. S-> aABb
2. A -> aAc
3. A -> λ
4. B -> bB
5. B -> c

และตาราง Parsing Table ต่อไปนี้

	a	b	c	$
A	2	3	3	
B		4	5	
S	1			

จงแสดงวิธีการหา derivation ของ acb โดยใช้ LL Parser

3. จาก Grammar

(1) E ⟶ E + T

(2) E ⟶ T

(3) T ⟶ T * F

(4) T ⟶ F

(5) F ⟶ (E)

(6) F -> id

และตารางต่อไปนี้

step	stack	Input	top of stack	token	rule
1	S $	acb $	S	a	1 (S → aABb)
2	a, A, B, b, $	acb $	a	a	Match terminal
3	A, B, b, $	cb $	A	c	3 (A → ℓ)
4	B, b, $	cb $	B	c	5 (B → c)
5	c, b, $	cb $	c	c	Match terminal
6	b, $	b $	b	b	Match terminal
7	$	$	$	$	complete

	Action						Goto		
State	id	+	*	(	)	$	E	T	F
0	S5		S4				1	2	3
1		S6				accept			
2		R2	S7		R2	R2			
3		R4	R4		R4	R4			
4	S5			S4			8	2	3
5		R6	R6		R6	R6			
6	S5			S4				9	3
7	S5			S4					10
8		S6			S11				
9		R1	S7		R1	R1			
10		R3	R3		R3	R3			
11		R5	R5		R5	R5			

จงแสดงการหา derivation ของ id * id โดยใช้ bottom up LR Parser

Stack	Input	Action
0	id+id * id $	shift 5
0id5	+ id * id $	Reduce 6 (use GOTO [0,F])
0F3	+ id * id $	Reduce 4 (use GOTO [0,T])
0T2	+ id * id $	Reduce 2 (use GOTO [0,E])
0E1	id * id $	shift 7
0E1+6	* id $	shift 5
0E1+6id5	* id $	Reduce 6 (use GOTO [6,F])
0E1+6F3	* id $	Reduce 4 (use GOTO [6,F])
0E1+6T9	* id $	shift 7
0E1+6T9*7	id $	shift 5
0E1+6T9*7id5	$	Reduce 6 (use GOTO [7,F])
0E1+6T9*7F10	$	Reduce 3 (use GOTO [6,T])
0E1+6T9	$	Reduce 1 (use GOTO [0,E])
0E1	$	Accept