C1) This is a recursive descent parser. Write the grammar from this parser.

```
block()
  match('{')
  stmt()
  match('}')

stmt()
  if( currenttoken == 'id')
    stmt1()
    stmt()

stmt1()
  match('id')
  match('=')
  expr()
  match(';')

expr()
  match('id')
  exprs()

exprs()
  if( currenttoken == '+')
    match('+')
    exprs()
```

block → { stmt }

stmt → stmt1 stmt | λ

stmt1 → id = expr ;

expr → id exprs

exprs → + exprs | λ

C2) Given this grammar, compute First and Follow set, draw the parsing table

```
dcl = ID dcl2
dcl2 =  ( formal ) stmt |  [ NUM ]
formal = ID formals | empty
formals = , formal | empty
```

	First	Follow
dcl	ID	$
dcl2	([	$
formal	ID λ	)
formals	, λ	)

1. dcl → ID dcl2
2. dcl2 → (formal) stmt
3. dcl2 → [NUM]
4. formal → ID formals
5. formal → λ
6. formals → , formal
7. formals → λ

	ID	(	)	[	,	$
dcl	1					
dcl2		2		3		
formal	4		5			
formals			7		6	