

Problem 1

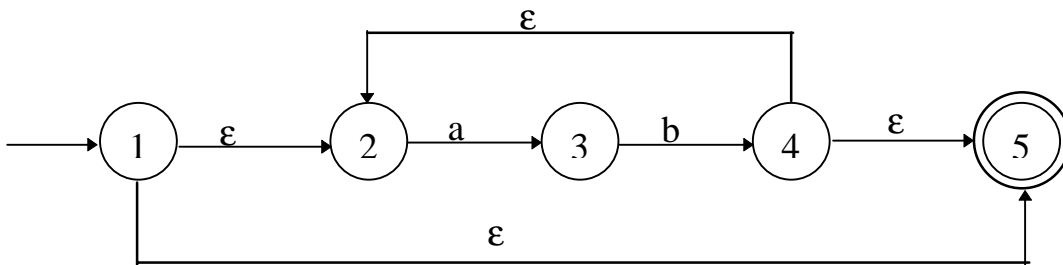
Part A: S can derive the empty string or $S \Rightarrow^* \epsilon$.

Part B: bison is a tool for automatically building a parser from a grammar.

Part C: “(a|b|c|\\|@)*”

Problem 2

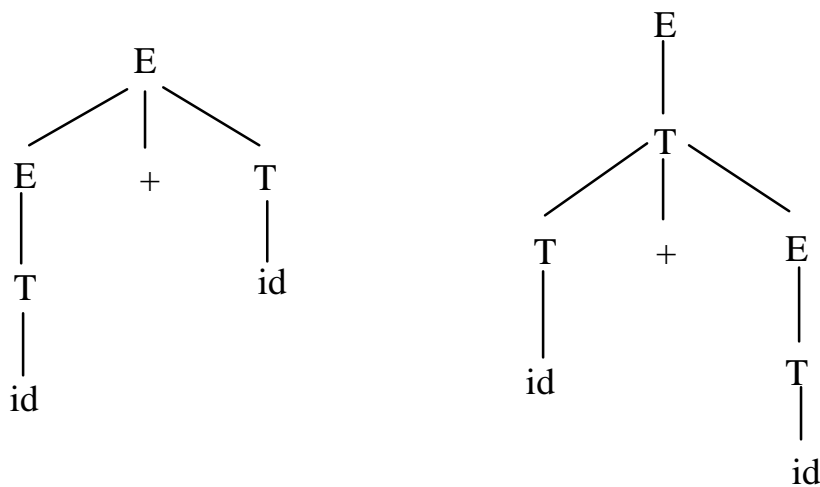
Part A:



Part B:

	a	b
[125]	[3]	[]
[3]	[]	[245]
[245]	[3]	[]

Problem 3



Problem 4

Part A:

(a) $S \rightarrow \text{if} (\text{expr}) S .$

(b) $S \rightarrow \text{if} (\text{expr}) S . \text{ else } S$

On this state, the parser can either shift **else** or reduce by production 1).

Part B: shift

Problem 5

Part A:

$E \rightarrow \text{id } Q$

$Q \rightarrow E + Q$

| $E - Q$

| $E * Q$

| E / Q

| ϵ

Part B:

$E \rightarrow \text{id } Q$

$Q \rightarrow E R$

| ϵ

$R \rightarrow + Q$

| $- Q$

| $* Q$

| $/ Q$

Part C:

$\text{First}(E) = \{\text{id}\}$

$\text{First}(Q) = \{\text{id}, \epsilon\}$

$\text{First}(R) = \{+, -, *, /\}$

$\text{Follow}(E) = \{+, -, *, /, \$\}$

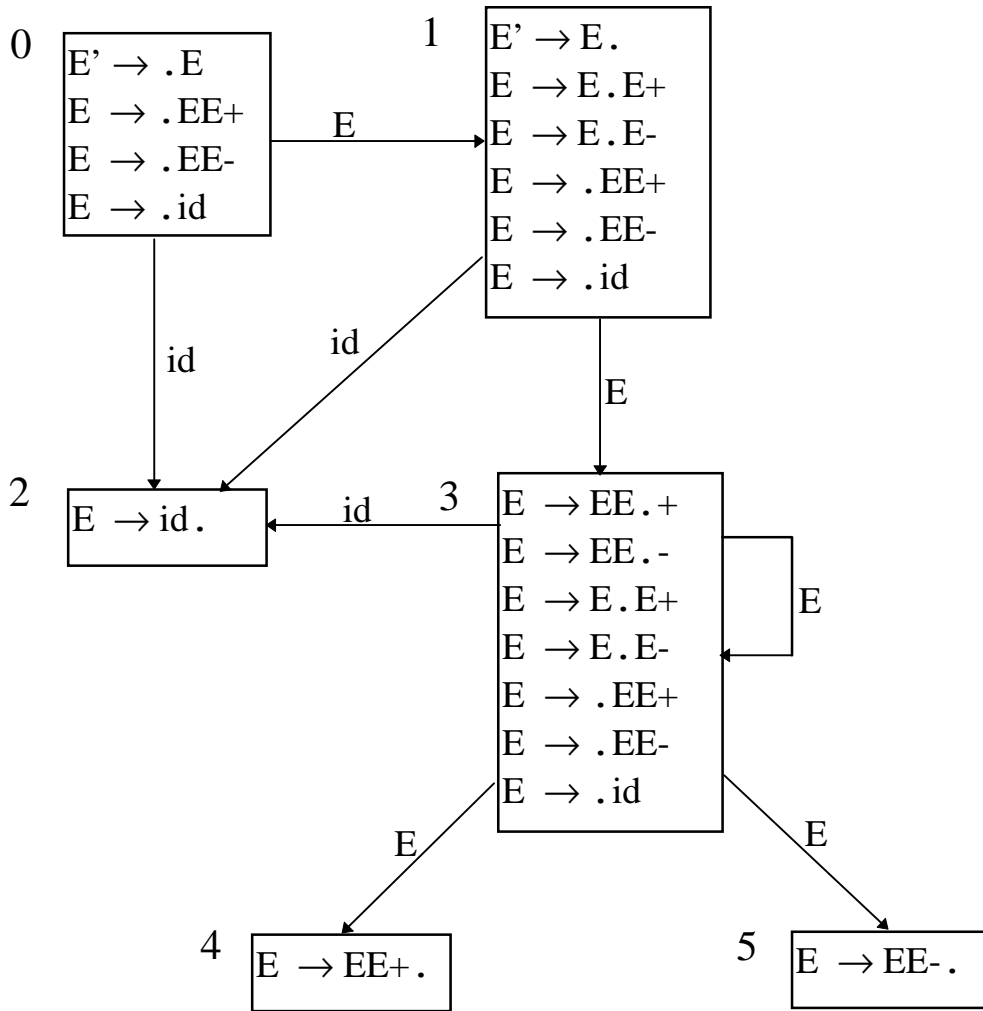
$\text{Follow}(Q) = \{+, -, *, /, \$\}$

$\text{Follow}(R) = \{+, -, *, /, \$\}$

Part D:

	id	+	-	*	/	\$
E	idQ					
Q	ER	ϵ	ϵ	ϵ	ϵ	ϵ
R		+ Q	- Q	* Q	/ Q	

Problem 6



Problem 7

SwStmt \rightarrow **switch** (**expr**) Stmt

Stmt \rightarrow **expr**

| SwStmt

| **default**: Stmt1

| **while** (**expr**) Stmt1

| { Slist }

SList \rightarrow ϵ

| Stmt ; SList1

Sw.legal=Stmt.num<=1;

Stmt.num=0

Stmt.num=0

Stmt.num=Stmt1.num+1;

Stmt.num=Stmt1.num

Stmt.num=Slist.num

SList.num=0

SList.num=Stmt.num+SList1.num