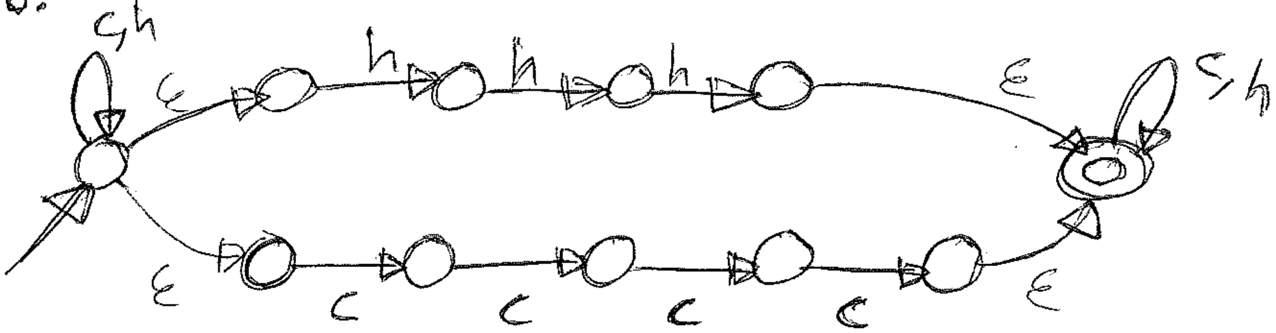
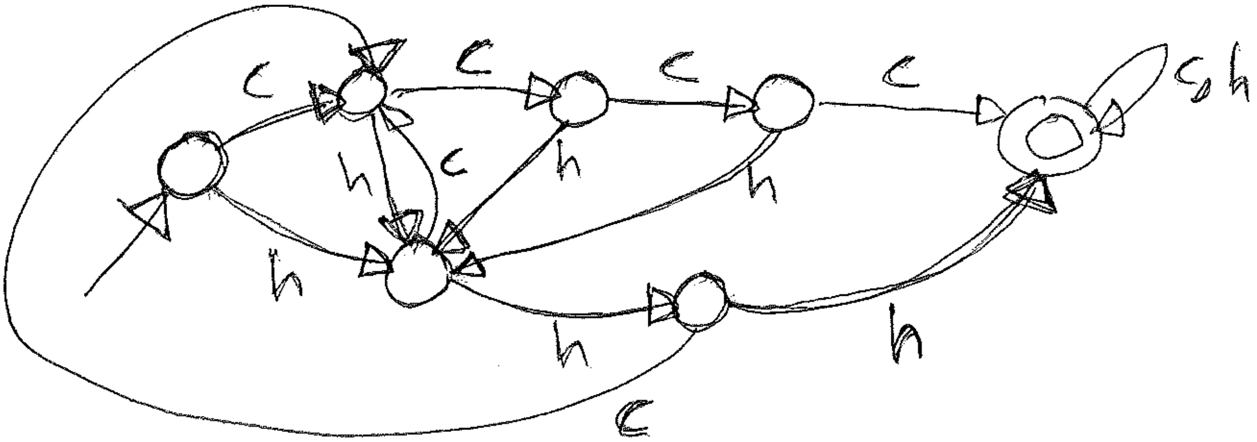
1. a. $(c|h)^*(hhh|cccc)(c|h)^*$

b.



c.



d. $\{a^n b^n a^n \mid n \geq 0\}$

5. a. $S \to S1 \mid AB2 \mid AB3$

$A \to A4 \mid 5$

$B \to B6 \mid \varepsilon$

Left recursion:

$A \to A4 \quad B \to B6$

$S \to S1$

Left factoring problem:

$S \to AB2 \mid AB3$

New grammar:

$S \to AB S_2 S_3$

$S_2 \to 2 \mid 3$

$S_3 \to 1 S_3 \mid \varepsilon$

$A \to 5 A_2$

$A_2 \to 4 A_2 \mid \varepsilon$

$B \to 6B \mid \varepsilon$

```
func S() {
  A(); B(); S2(); S3();
}
```

```
func S2() {
  if (token==2)
     scan();
  else if (token==3)
     scan();
  else
     error(...);
  end if
}
```

```
func S3() {
  if (token ==1)
     scan();
     S3();
  end if;
}
```

```
func A() {
  if (token == 5)
     scan ();
     A2();
  else
     error(...);
  end if
}
```

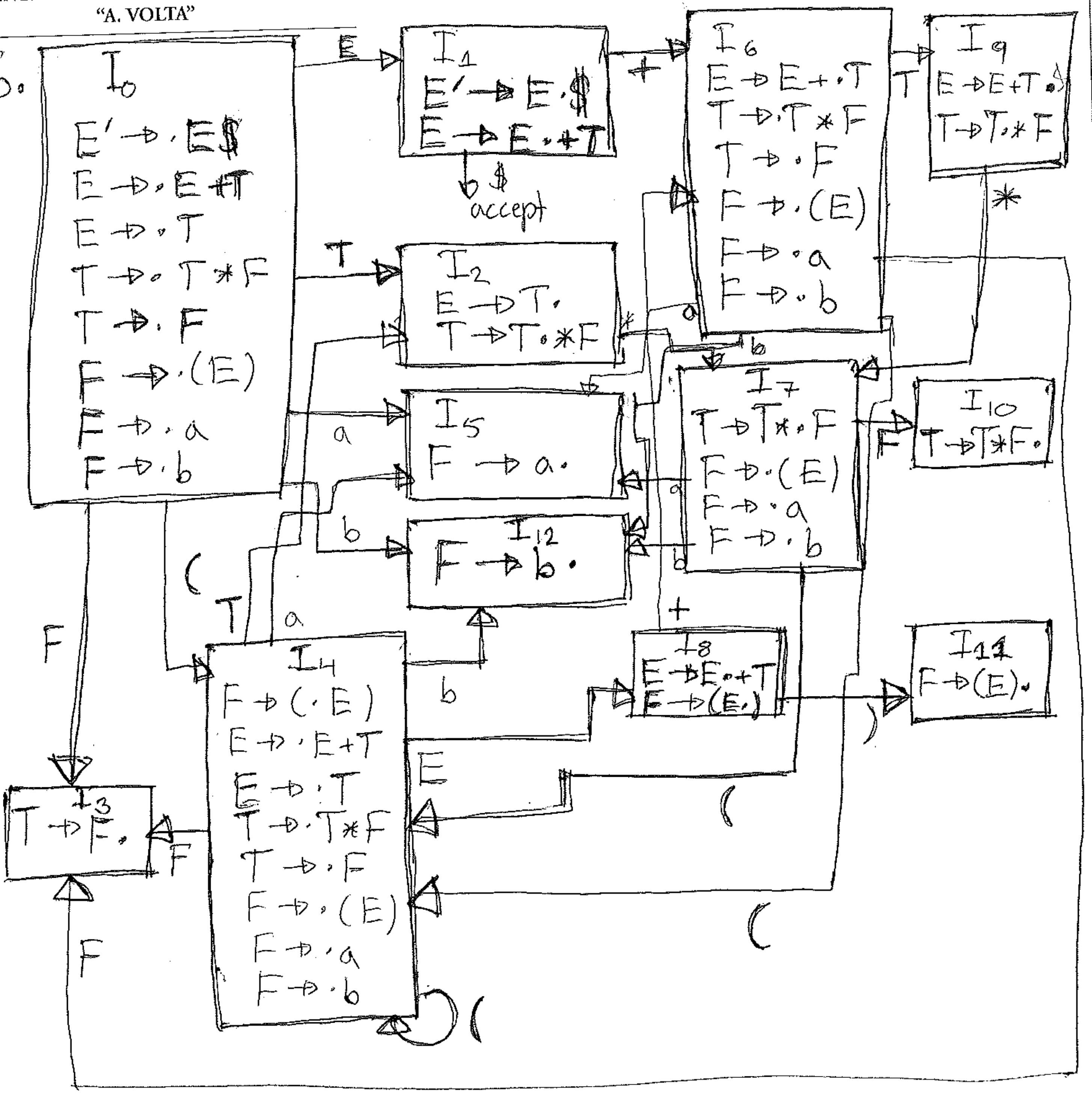
```
func main() {
  scan();
  S();
  if (not eof)
     error(...);
  end if;
}
```

```
func A2() {
  if (token ==4)
     scan(); A2();
  end if;
}
```

```
func B() {
  if (token==6)
     scan(); B();
  end if
}
```

b. The program stack that is used in the different function calls.

6.



Shift-reduce conflicts in , I2 and I9.

FOLLOW(E') = { }
FOLLOW(E) = {$,), +}
FOLLOW(T) = {$,), *, +}
FOLLOW(F) = {$,), +, *}

But * not in FOLLOW(E)

[redacted]

Thus SLR(1).

CENTRO DI CULTURA SCIENTIFICA
"A. VOLTA"

(1) E → E+T
(2) E → T
(3) T → T*F
(4) T → F
(5) F → (E)
(6) F → a
(7) F → b

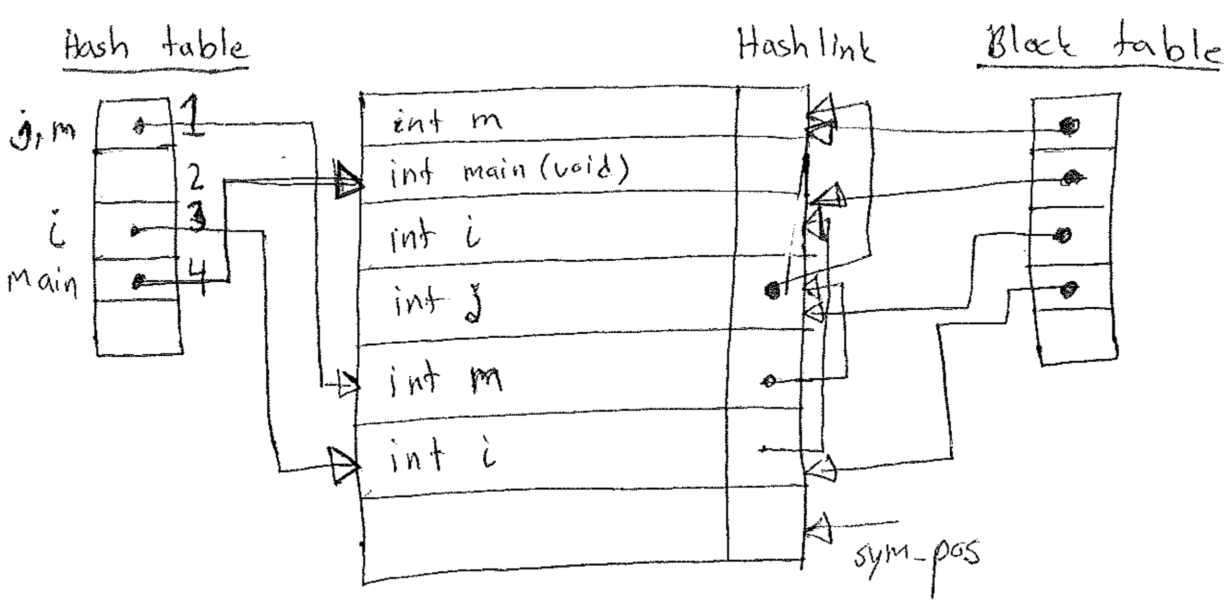
6. continued.

State	a	b	+	*	(	)	$	E	T	F
0	s5	s12			s4			1	2	3
1			s6				acc			
2			r2	s7		r2	r2			
3			r4	r4		r4	r4			
4	s5	s12			s4			8	2	3
5			r6	r6		r6	r6			
6	s5	s12			s4				9	3
7	s5	s12			s4					10
8			s6			s11				
9			r1	s7		r1	r1			
10			r3	r3		r3	r3			
11			r5	r5		r5	r5			
12			r7	r7		r7	r7			

0	a*b+b*(a+a)$
0a5	*b+b*(a+a)$
0F3	*b+b*(a+a)$
0T2	*b+b*(a+a)$
0T2*7	b+b*(a+a)$
0T2*7b12	+b*(a+a)$
0T2*7F10	+b*(a+a)$
0T2	+b*(a+a)$
0E1	+b*(a+a)$
0E1+6	b*(a+a)$
0E1+6b12	*(a+a)$
0E1+6F3	*(a+a)$
0E1+6T9	*(a+a)$
0E1+6T9*7	(a+a)$
0E1+6T9*7(4	a+a)$
0E1+6T9*7(4a5	+a)$
0E1+6T9*7(4F3	+a)$
0E1+6T9*7(4T2	+a)$
0E1+6T9*7(4E8	+a)$
0E1+6T9*7(4E8+6	a)$
0E1+6T9*7(4E8+6a5	)$
0E1+6T9*7(4E8+6F3	)$
0E1+6T9*7(4E8+6T9	)$
0E1+6T9*7(4E8	)$
0E1+6T9*7(4E8)11	$
0E1+6T9*7F10	$
0E1+6T9	$
0E1	$
ACCEPT!	

Symbol table

7.



For looking up m in $i = m * 2$ we simply hash m, get an entry to the hash table, follow the link and we find the m we were looking for. If we hadn't found correct m we would have had to follow hash/back links.

Use the block table to get to the beginning of the current block and then enumerate the variables from there.

8. Rewrite the grammar to:

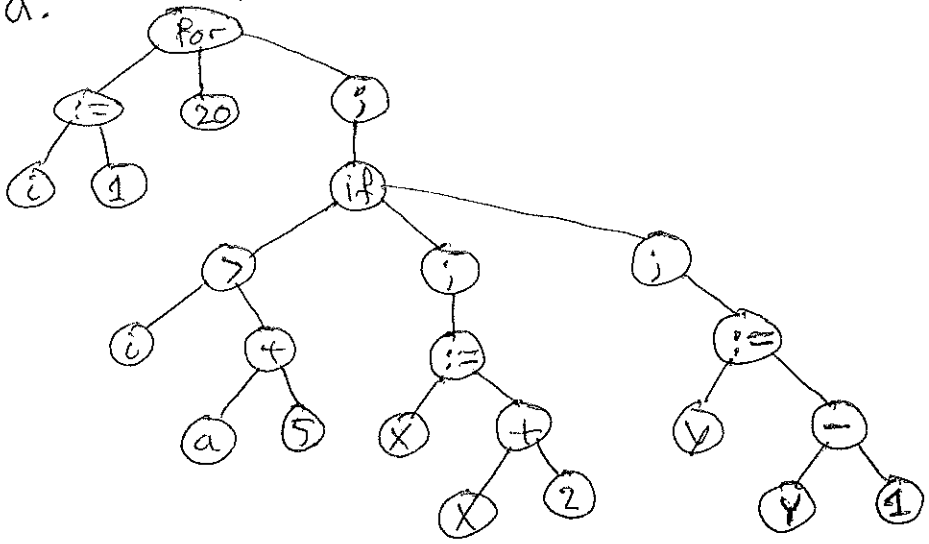
```
<loop> ::= for <id-range> loop <statement_list>
                                  end for;
<id-range> ::= <id> in <start expr> : <stepexpr> : <endexpr>
```

```
<loop> ::= for   <id-range > loop <statement_list> end for;
{   GEN(JMP, <id_range>.addr, 0, 0);
    QUADRUPLE[<id-range>.quad][1] = current_quad + 1;  }

<id_range> ::=  <id> in <start_expr> : <step_expr> : <end_expr>
{   GEN(ASSIGN, <start_expr>.addr, 0, <id>.addr );
    int temp = current_quad + 1;
    GEN (JMP, 0, 0, 0);
    <id_range>.addr = current_quad + 1;   // We can make use of the addr field as well
    int temp_var = gentemp();
    GEN (ADD, <id>.addr , <step_expr>.addr, temp_var);
    GEN (ASSIGN, temp_var, 0, <id>.addr);
    QUADRUPLE[temp][1] = current_quad + 1;
    temp_var = gentemp();
    GEN( GTEQ , <end_expr>.addr, <id>.addr, temp_var);
    <id_range>.quad = current_quad + 1;
    GEN(JMPF, 0, temp_var, 0);
}
```

<id-range> will be reduced first, then <loop> (in a bottom-up parser).

9. a. Abstract Syntax Tree



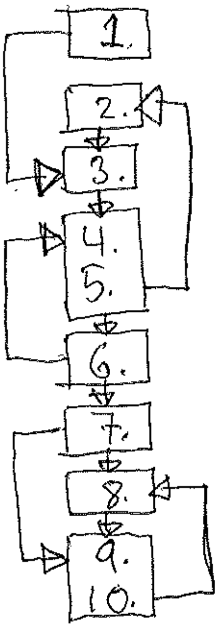
Quadruples

1. qassign	1	—	i
2. qlabel	1	—	—
3. qgteq	20	i	t_1
4. qjmpt	4	t_1	—
5. qadd	a	5	t_2
6. qgt	i	t_2	t_3
7. qjmpf	2	t_3	—
8. qadd	x	2	t_4
9. qassign	t_4	—	x
10. qjmp	3	—	—
11. qlabel	2	—	—
12. qsub	y	1	t_5
13 qassign	t_5	—	y
14. qlabel	3	—	—
15. qadd	i	1	t_6
16. qassign	t_6	—	i
17. qjmp	1	—	—
18. qlabel	4	—	—

Postfixcode

```
       i 1 :=
L1  20 i >=  L4 JEQZ
       i a 5 + >  L2 JEQZ
       x x 2 + := L3 JMP
L2     y y 1 - :=
L3     i i 1 + := L1 JMP
L4
```

```
1    goto L2
2    L1: x := x+1
3    L2: x := x+1
4.   L3: x := x+1
5.   if x=1 then goto L1
6.   if x=2 then goto L8
7.   if x=3 then goto L5
8.   L4: x := x+1
9.   L5: x := x+1
10.  if x=4 then goto L4
```


1.
2.
3.
4.
5.
6.
7.
8.
9.
10.