1. Grammar given.
   a. The given grammar isn't LL(1) because the CMD var has common prefix 'printf' for more than one rule.
   b. The fixed grammar:

$$PROG \rightarrow TDEF\ BLK$$
$$TDEF \rightarrow \text{'var' 'is' 'type' ';'}\ TDEF \mid \varepsilon$$
$$BLK \rightarrow \text{'\{'}\ STL\ \text{'\}'}$$
$$STL \rightarrow CMD\ STL \mid \varepsilon$$
$$CMD \rightarrow \text{'printf'}\ P \mid \text{'var' '=' 'num' ';'} \mid BLK$$
$$P \rightarrow \text{'num' ';'} \mid \text{'var' ';'}$$

   c. Let's show that the fixed grammar is LL(1).
      i. We'll start by building the FIRST table:

PROG	TDEF	BLK	STL	CMD	P
-	-	-	-	-	-
-	'var'	'{'	-	'printf', 'var', '{'	'num', 'var'
'var' , '{'	'var'	'{'	'printf', 'var', '{'	'printf', 'var', '{'	'num', 'var'
'var' , '{'	**'var'**	**'{'**	**'printf', 'var', '{'**	**'printf', 'var', '{'**	**'num', 'var'**

      ii. Next we'll build the FOLLOW table:

PROG	TDEF	BLK	STL	CMD	P
$	-	-	-	-	-
$	'{'	-	'}'	'printf', 'var', '{', '}'	-
$	'{'	-	'}'	'printf', 'var', '{', '}'	-

      iii. And finally we'll calculate SELECT for each rule. For each variable, if the all SELECTs have no terminal in common then the grammar is LL(1):

$$SELECT(PROG \rightarrow TDEF\ BLK) = \text{'var'}$$
$$SELECT(TDEF \rightarrow \text{'var' 'is' 'type' ';'}\ TDEF) = \text{'var'}\ \#SELECT(TDEF \rightarrow \varepsilon) = \text{'\{'}$$
$$SELECT(BLK \rightarrow \text{'\{'}\ STL\ \text{'\}'}) = \text{'\{'}$$
$$SELECT(STL \rightarrow CMD\ STL) = \text{'printf', 'var', '\{'}\ \#SELECT(STL \rightarrow \varepsilon) = \text{'\}'}$$
$$SELECT(CMD \rightarrow \text{'printf'}\ P) = \text{'printf'}\ \#SELECT(\text{'var' '=' 'num' ';'}) = \text{'var'}\ \#SELECT(BLK) = \text{'\{'}$$
$$SELECT(P \rightarrow \text{'num' ';'}) = \text{'num'}\ \#SELECT(\text{'var' ';'}) = \text{'var'}$$

d. Let's assign number to all rules:

1 $PROG \rightarrow TDEF\ BLK$

2 $TDEF \rightarrow$ 'var' 'is' '$type$' ';' $TDEF$

3 $TDEF \rightarrow \varepsilon$

4 $BLK \rightarrow$ '{' STL '}'

5 $STL \rightarrow CMD\ STL$

6 $STL \rightarrow \varepsilon$

7 $CMD \rightarrow$ 'printf' P

8 $CMD \rightarrow$ 'var' ' = ' 'num' ';'

9 $CMD \rightarrow BLK$

10 $P \rightarrow$ 'num' ';'

11 $P \rightarrow$ 'var' ';'

	'{'	'}'	'printf'	'var'	'num'
PROG				1	
TDEF	3			2	
BLK	4				
STL	5	6	5	5	
CMD	9		7	8	
P				10	11

2. See the state machine description below.
   a. The given grammar is LR(1) because the parser described by the state machine below has no conflicts.
      For each state we can easily decide weather we need to ‘shift’ or ‘reduce’.
   b. However, the grammar is NOT LR(0). Let’s look at State4: if the current token is ‘m’ than we may perform a ‘shift-6’ operation. But, bereft of a lookahead, we may also perform a ‘reduce’ operation with the rule $H \rightarrow k$.
   c. The same example is enough to show that the grammar is not SLR either. Because ‘m’ is clearly in FOLLOW(H), we would still want to perform the ‘reduce’ operation AND the ‘shift’ operation.
   d.

	Operations Table				**GOTO Table**	
	m	**k**	**d**	**$**	**A**	**H**
0		s5	s2		1	3
1				acc		
2		r4				
3		s5				
4	s6	r3				
5		s7	s8			9
6		s11	s2		13	12
7	r3					
8	r4					
9	s10					
10				r2		
11	s16					
12		s15				
13			s14			
14				r1		
15		s7	s8			19
16		s11			17	
17			s18			
18			r1			
19	s20					
20			r2			

Rules index: (0) $S \rightarrow A$

(1) $A \rightarrow kmAd$ (2) $A \rightarrow HkHm$

(3) $H \rightarrow k$ (4) $H \rightarrow d$

2. A list of states:

State0:

$S \to \bullet A, \$$

$A \to \bullet kmAd, \$$

$A \to \bullet HkHm, \$$

$H \to \bullet k, k$

$H \to \bullet d, k$

State1:

$S \to A \bullet, \$$

State2:

$H \to d \bullet, k$

State3:

$A \to H \bullet kHm, \$$

State4:

$A \to k \bullet mAd, \$$

$H \to k \bullet, k$

State5:

$A \to Hk \bullet Hm, \$$

$H \to \bullet k, m$

$H \to \bullet d, m$

State6:

$A \to km \bullet Ad, \$$

$A \to \bullet kmAd, d$

$A \to \bullet HkHm, d$

$H \to \bullet k, k$

$H \to \bullet d, k$

State7:

$H \to k \bullet, m$

State8:

$H \to d \bullet, m$

State9:

$A \to HkH \bullet m, \$$

State10:

$A \to HkHm \bullet, \$$

State11:

$A \to k \bullet mAd, d$

$H \to \bullet k, k$

State12:

$A \to H \bullet kHm, d$

State13:

$A \to kmA \bullet d, \$$

State14:

$A \to kmAd \bullet, \$$

State15:

$A \to Hk \bullet Hm, d$

$H \to \bullet k, m$

$H \to \bullet d, m$

State16:

$A \to km \bullet Ad, d$

$A \to \bullet kmAd, d$

$A \to \bullet HkHm, d$

$H \to \bullet k, k$

$H \to \bullet d, k$

State17:

$A \to kmA \bullet d, d$

State18:

$A \to kmAd \bullet, d$

State19:

$A \to HkH \bullet m, d$

State20:

$A \to HkHm \bullet, d$

A list of transitions:

From **State0**:

Simbol	**State**
A	State1
d	State2
H	State3
k	State4

From **State3**:

Simbol	**State**
k	State5

From **State4**:

Simbol	**State**
m	State6

From **State5**:

Simbol	**State**
H	State9
k	State7
d	State8

From **State6**:

Simbol	**State**
k	State11
d	State2
H	State12
A	State13

From **State9**:

Simbol	**State**
m	State10

From **State11**:

Simbol	**State**
m	State16

From **State12**:

Simbol	**State**
k	State15

From **State13**:

Simbol	**State**
d	State14

From **State15**:

Simbol	**State**
k	State7
d	State8
H	State19

From **State16**:

Simbol	**State**
k	State11
A	State17

From **State17**:

Simbol	**State**
d	State18

From **State19**:

Simbol	**State**
m	State20