Solution to Homework 2:
LL(1) parsing

In order for the grammar to parse the terminal symbols in the correct order, we introduce new non-terminals and produce the following grammar:

Grammar (1)

```
S       ::= Exp $

Exp     ::= Exp || AndAnd
          | AndAnd

AndAnd  ::= AndAnd && Or
          | Or

Or      ::= Or | And
          | And

And     ::= And & Not
          | Not

Not     ::= ! Not
          | Unit

Unit    ::= true
          |  false
```

Our next step would be to eliminate the left recursion which the LL(1) parser can not handle, so we produce our final grammar(2):

Grammar (2)

S ::= Exp $

Exp ::= AndAnd Exp'
Exp' ::= || AndAnd Exp'
| ε

AndAnd ::= Or AndAnd'
AndAnd' := && Or AndAnd'
| ε

Or ::= And Or'
Or' ::= | And Or'
| ε

And ::= Not And'
And' := & Not And'
| ε

Not ::= ! Not
| Unit

Unit ::= true
| false

The next step in the creation of an LL(1) parser is to calculate the FIRST and FOLLOW sets:

Non-terminal	Nullable	FIRST	FOLLOW
S		!, true, false	
Exp		!, true, false	$
Exp’	Yes	||, ε	$
AndAnd		!, true, false	||, $
AndAnd’	Yes	&&, ε	||, $
Or		!, true, false	&&, ||, $
Or’	Yes	|, ε	&&, ||, $
And		!, true, false	|, &&, || ,$
And’	Yes	&, ε	|, &&, ||, $
Not		!, true, false	&, |, &&, ||, $
Unit		true, false	&, |, &&, ||, $

Now we are ready to construct the *predictive parsing* table

	!	&	|	&&	||	true	False	$
S	S->Exp $					S->Exp $	S->Exp $	
Exp	Exp -> AndAnd Exp’					Exp -> AndAnd Exp’	Exp -> AndAnd Exp’	
Exp’					Exp’-> ||AndAnd Exp’			Exp->ε
AndAnd	AndAnd-> Or AndAnd’					AndAnd-> Or AndAnd’	AndAnd-> Or AndAnd’	
AndAnd’				AndAnd’-> && Or AndAnd’	AndAnd’->ε			AndAnd’-> ε
Or	Or->And Or’					Or->And Or’	Or->And Or’	
Or’			Or’-> |And Or’	Or’->ε	Or’->ε			Or’->ε
And	And-> Not And’					And-> Not And’	And-> Not And’	
And’		And’-> & Not And’	And’->ε	And’->ε	And’->ε			And’->ε
Not	Not->! Not					Not -> Unit	Not -> Unit	
Unit						Unit-> true	Unit->false	

What is left is just to code the above table in a LL(1) parser