For this homework assignment, submit your answers on paper at the start of the lecture on the due date.

Late homework assignments will not be accepted, unless you have a valid written excuse (medical, etc.). You must do this assignment alone. No team work or "talking with your friends" will be accepted. No copying from the Internet. Cheating means zero.

Consider the following context-free grammar for simple bitwise expressions in the C programming language:

```
B -> T | B ^ T
T -> F | T & F
F -> ~ F | ( B ) | 1 | 0
```

where '^' (the C bitwise "xor" operator), '&' (the C bitwise "and" operator), '~' (the C bitwise negation operator), '(', ')', '1', and '0' are tokens, and B is the start symbol for the grammar.

Note: this grammar does not contain the '|' C bitwise "or" operator so that you do not confuse it with the symbol | that we use to separate grammar productions.

1) Using the grammar above, construct a leftmost derivation and a parse tree for the following input:

~ (1 ^ 0)

2) Eliminate left-recursion from the grammar (if necessary) and left-factor the grammar (if necessary). Explain step by step how you do it.

3) Use the new grammar from Question 2 to construct a leftmost derivation and a parse tree for the same input as in Question 1.

4) Compute the FIRST and FOLLOW sets for the new grammar from Question 2. Explain step by step how you do it.

Note: for FIRST, list the tokens in the following order from left to right:
1, 0 , ~, ^, &, (,)
For FIRST and FOLLOW, list the nonterminals in the following order from left to right: B, B', T, T', F

5) Use the FIRST and FOLLOW sets computed in Question 4 to create a parsing table for a top-down non-recursive predictive parser.
Explain step by step how you do it.

Note: for the columns and rows, list the tokens and nonterminals in the same order as in Question 4.

Note: there is no need to do any error recovery in the parsing table.

6) Use the parsing table from Question 5 to parse the input from Question 1. Show the content of the stack at each step and the output of the parser. There is no need to explain each step.

7) Use the output computed by the parser in Question 6 to create a derivation and a parse tree for the parser's input. Compare the result with your answer to Question 3.