CSCI-621 Programming Languages

Programming Assignment 3

A Non-recursive Predictive Parser

Design and implement a Non-recursive Predictive Parser (NPP) for the following grammar:

<elist> → <elist> , <e> | <e>
<e> → <n> ^ <e> | <n>
<n> → <n> <d> | <d>
<d> → 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

Where ^ is an exponentiation operator (associate to right). This grammar generates statements of the form 2^2^3, 15, 20^2 for which the parser outputs 256 15 400.

Solution:
In order to implement this NPP, a parse table must be constructed by using first sets and follow sets.

The parse table for this parser is shown as follows:

	d	^	**,**	**$**
elist	**elist→ e elist'**			
elist'			**elist' → , elist**	**elist' → ε**
e	e → n e'			
e'		e' → ^ e	e' → ε	e' → ε
n	n → d n'			
n'	n' →n	n' → ε	n' → ε	n' → ε

The Non-recursive Predictive Parsing Algorithm is:

Let T$ be the input string followed by a $.
Set ip to point to the first symbol of T$.
Repeat
    Let X be the top of stack symbol.
    Let **a** be the symbol pointed to by ip.
    IF X is a terminal or $ THEN
    IF X== **a** THEN
    Pop X from the stack and advance ip
    ELSE error()
ELSE IF M[X,**a**] == $X \rightarrow Y_1 Y_2 ... Y_k$ THEN
    BEGIN
        pop X from the stack
        push $Y_k, Y_{k-1}, ..., Y_1$ onto to the stack with Y_1 on the top
        output $X \rightarrow Y_1 Y_2 ... Y_k$
    END
ELSE error()
UNTIL X == $