CD Assignment-3 Report

Name: Utkarsh Aditya
Roll No: S20180010182

Overview :

Implementation of SDT along with parsing the given input string and producing the output.

Files:> 1) lex.py - It returns the token stream(list) of tokens for the given input string.
2) parser.py - It has the CLR parser logic along with SDT logic. For a given input it will output the parsing steps along with all the details of each step(stack status, action name, attribute calculation, etc)
3) Parse_table.csv - It has the parse table for the final grammar which is being used in the assignment.
4) CLR_Parse_Table.xlsx - For task 5 of the assignment.
5) automata.svg - For task 4 of the assignment.

Task Breakdown:

1) Definition of number in lex will be:
   num ⇒ [0-9]+

2) Modified SDT with ^ operator:

L⇒E n	{ print(E.val); }
E⇒E+T	{ E.val = E.val+T.val; }
E⇒T	{ E.val = T.val; }
T⇒T*F	{ T.val = T.val*F.val; }
T⇒F	{ T.val = F.val; }
F⇒F^A	{ F.val = F.val^A.val; }
F⇒A	{ F.val = A.val; }
A⇒(E)	{ A.val = E.val; }
A⇒num	{ A.val = num.lexval; }

3) Final Modified SDT with unary minus operator

L⇒E n	{ print(E.val); }
E⇒E1+T	{ E.val = E1.val+T.val; }
E⇒T	{ E.val = T.val; }
T⇒T1*F	{ T.val = T1.val*F.val; }
T⇒F	{ T.val = F.val; }
F⇒F1^A	{ F.val = F1.val^A.val; }
F⇒A	{ F.val = A.val; }
A⇒-A1	{ A.val = -A1.val; }
A⇒(E)	{ A.val = E.val; }
A⇒num	{ A.val = num.lexval; }

4) LR1 Automaton is in automata.svg
Steps : A) Begin with the base LR(1) item L⇒.E,$
B) Inductively find closure of the kernel of the LR(1) item and apply GOTO.
C) Repeat until no more command can be added.

5) The CLR parse table is in the CLR_Parse_Table.xlsx file.

6) SDT along with parsing the given input string has been implemented in parser.py

Commands to run:

1)
```
python .\parser.py
```

Result

Output is provided in png and txt format with 2 different examples.
“SDT CALCULATION” shows the SDT steps in the output.