COMPILER DESIGN

EXP – 11 Shift Reduce Parsing

Tamojit Sarkar
RA1811027010034
CSE-BD Sec-I2

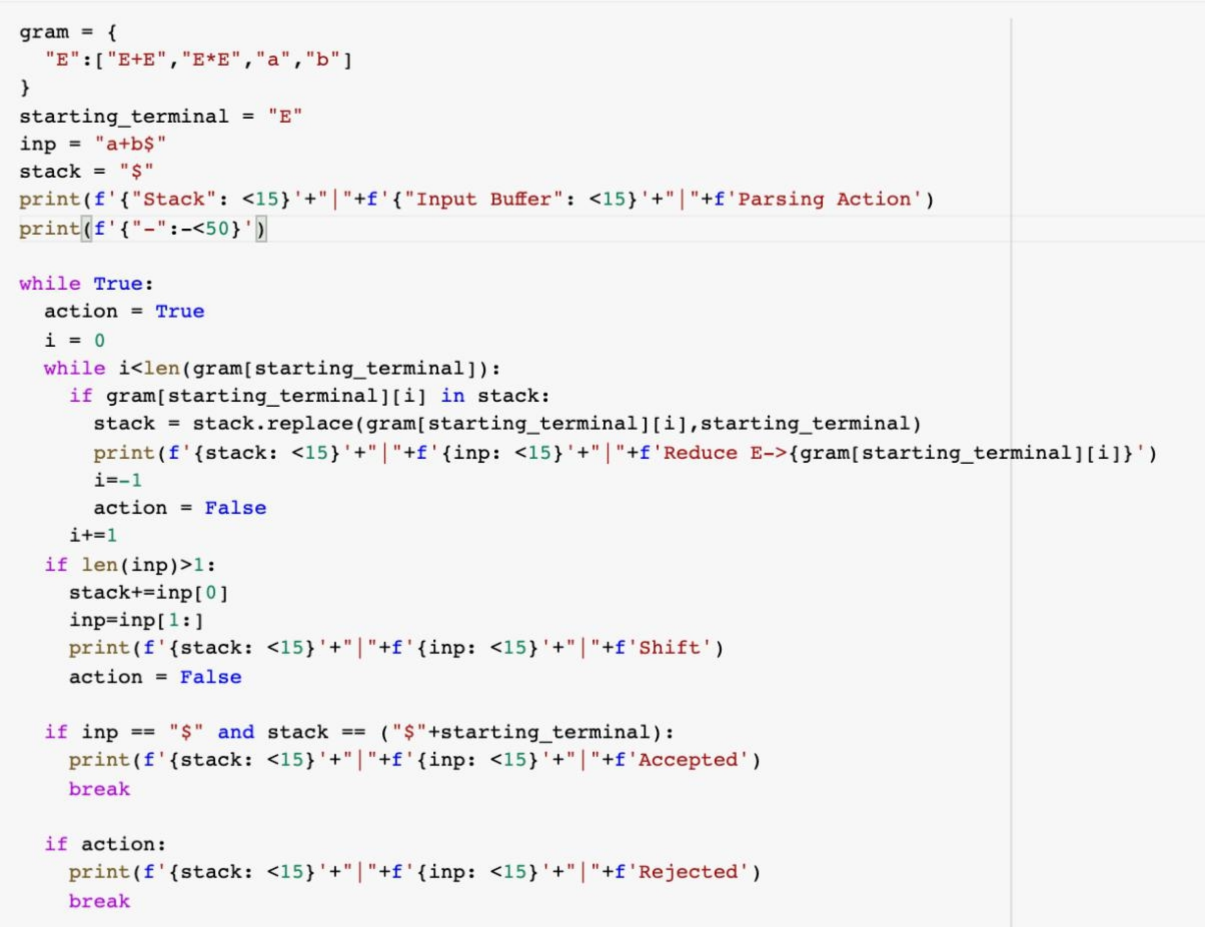
Aim: To compute intermediate code generation –infix to prefix and postfix

Language Used: Python

Algorithm:

1. Declare array for string and stack and other necessary variables.
2. Get the expression from the user and store it as string.
3. Append $ to the end of the string.6. Store $ into the stack.
4. Print three columns as “Stack”, “Input Symbol” and “Action” for the respective actions.
5. Use for loop from i as 0 till string length and check the string.
6. If string has some operator or id, push it to the stack.
7. Mark this action as “Shift‟.
8. Print the stack, string and action values.
9. If stack contains some production on shifting, reduce it.
10. Mark this action as “Reduce‟.
11. Print the stack, string and action values.
12. Repeat steps 6 to 10 again and again till the for loop is valid.
13. Now check the string and the stack.
14. If the string contains only $ and the stack has only $E within it, then print that
    the “Accepted”.
15. Else print that the “Rejected”.
16. End the program

Code:



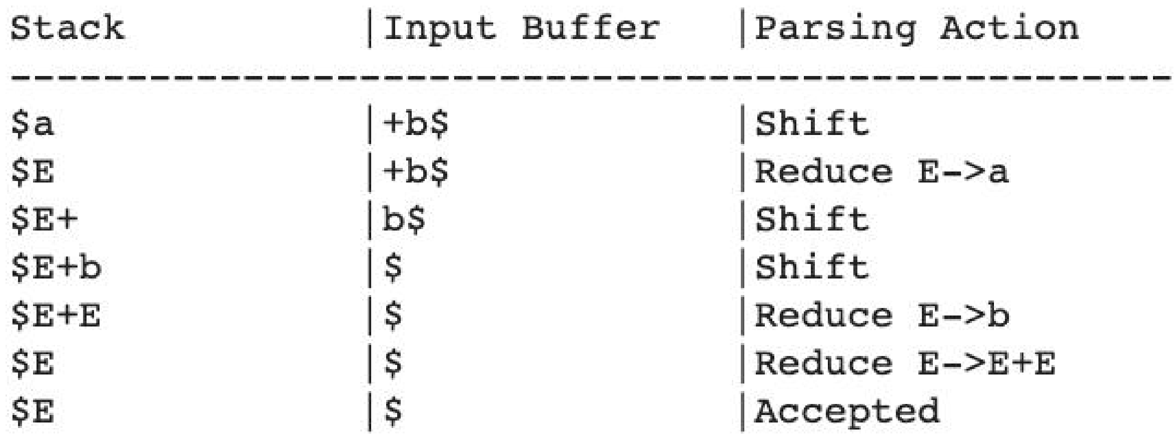
```python
gram = {
  "E":["E+E","E*E","a","b"]
}
starting_terminal = "E"
inp = "a+b$"
stack = "$"
print(f'{"Stack": <15}'+"|"+f'{"Input Buffer": <15}'+"|"+f'Parsing Action')
print(f'{"-":-<50}')

while True:
  action = True
  i = 0
  while i<len(gram[starting_terminal]):
    if gram[starting_terminal][i] in stack:
      stack = stack.replace(gram[starting_terminal][i],starting_terminal)
      print(f'{stack: <15}'+"|"+f'{inp: <15}'+"|"+f'Reduce E->{gram[starting_terminal][i]}')
      i=-1
      action = False
    i+=1
  if len(inp)>1:
    stack+=inp[0]
    inp=inp[1:]
    print(f'{stack: <15}'+"|"+f'{inp: <15}'+"|"+f'Shift')
    action = False

  if inp == "$" and stack == ("$"+starting_terminal):
    print(f'{stack: <15}'+"|"+f'{inp: <15}'+"|"+f'Accepted')
    break

  if action:
    print(f'{stack: <15}'+"|"+f'{inp: <15}'+"|"+f'Rejected')
    break
```

Output:



```
Stack          |Input Buffer   |Parsing Action
--------------------------------------------------
$a             |+b$            |Shift
$E             |+b$            |Reduce E->a
$E+            |b$             |Shift
$E+b           |$              |Shift
$E+E           |$              |Reduce E->b
$E             |$              |Reduce E->E+E
$E             |$              |Accepted
```

Result:

Thus the Shift reduce parser has been successfully implemented.