EXP:7: IMPLEMENTATION OF SHIFT REDUCE PARSING

AIM: To design a code to implement shift reduce parsing.

LANGUAGE USED: Python 3

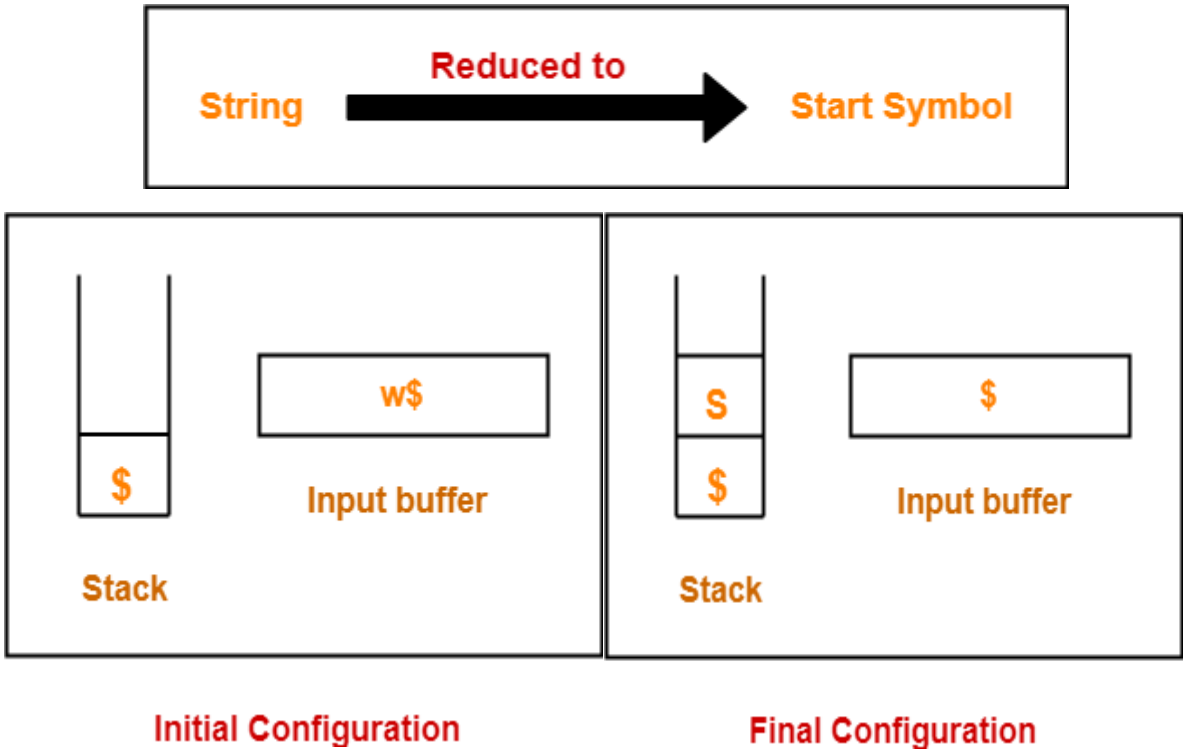
ALGORITHM/PROCEDURE: -

- Epsilon is represented by ‘e’.
- Productions are of the form A=B, where ‘A’ is a single Non-Terminal and ‘B’ can be any combination of Terminals and Non- Terminals.
- Each production of a non-terminal is entered on a different line.
- Only Upper-Case letters are Non-Terminals and everything else is a terminal.
- Do not use ‘!’ or ‘$’ as they are reserved for special purposes.
- Grammar is taken as input and will be through an infinite loop (while=true).
- And then by the rules of the shift reduce parsing each stack and input string will be verified by the productions and will end only if the grammar is accepted or when it is rejected.
- If the grammar stops at the starting symbol given for the production, then the string will get accepted.
- Or else if an error occurs, if any string is not present in the production after completing all the steps then the parsing will get rejected and will get exited from the loop.
- In this process, if the string matches any production in the LHS then it will perform the reduce action.
- In this process, if the string does not matches any production it will perform shift action until the string gets accepted or rejected.

EXPLANATION:

- Shift reduce parsing is a process of reducing a string to the start symbol of a grammar.
- Shift reduce parsing uses a stack to hold the grammar and an input tape to hold the string.
- Sift reduce parsing performs the two actions: shift and reduce. That's why it is known as shift reduces parsing.
- At the shift action, the current symbol in the input string is pushed to a stack.
- At each reduction, the symbols will be replaced by the non-terminals. The symbol is the right side of the production and non-terminal is the left side of the production.

SPACE TREE DIAGRAM/ EXPLANATION:

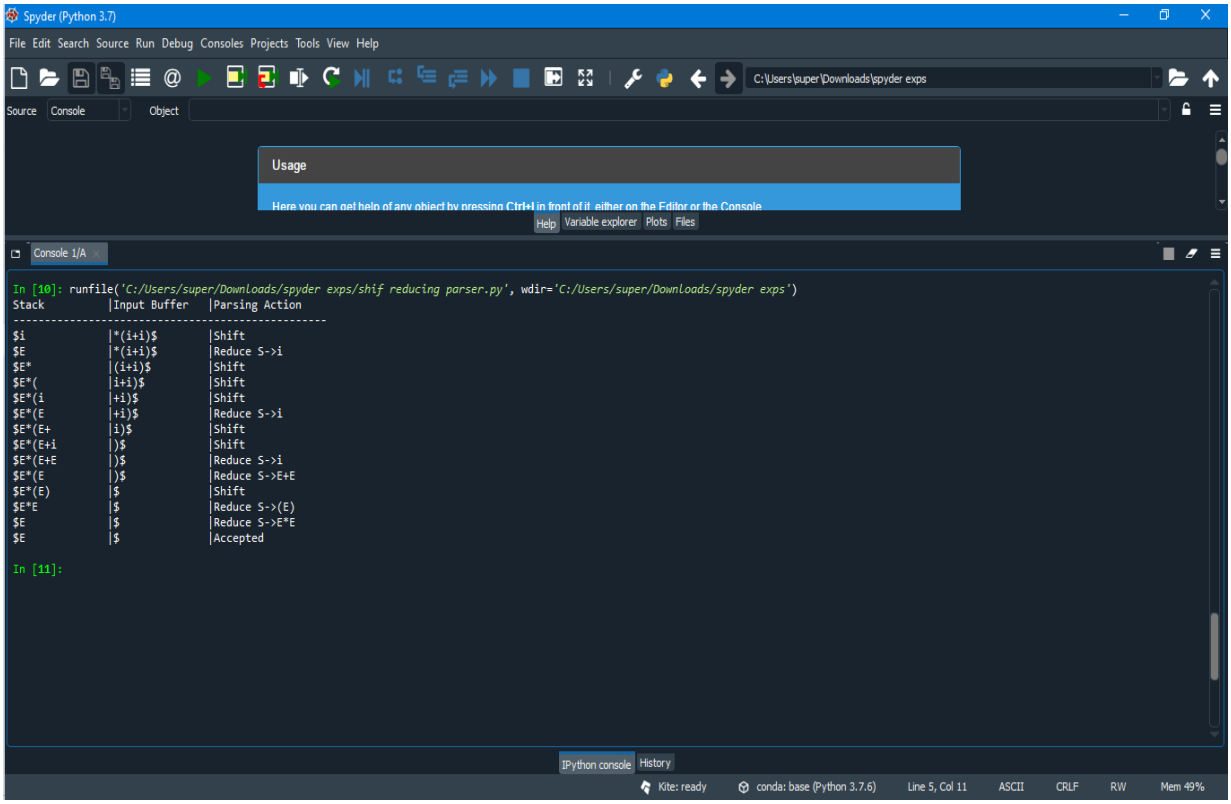


SOURCE CODE: -

```
gram = {
        "E":["E+E","E*E","(E)","i"]
}
starting_terminal = "E"
inp = "i*( i+i)$"
stack = "$"
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 S->{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:



RESULT: Therefore, we successfully implemented a code for shift reducing parser using python.