11/3/2022

## Experiment 7: Shift Reduce Parsing

**Aim:** To write a program to perform Shift Reduce Parsing.

**Algorithm:**

1. Start
2. Input is fed into the program in the form of three production rules: E → 2E2, E → 3E3, E → 4.
3. Copy input string to a global array.
4. For every character input from the global array, check against one of the following actions:

i) Shift: Shift the next input symbol onto the top of stack

ii) Reduce: If handle appears on top of stack then reduce it using proper production rule. RHS of rule is popped out of stack and LHS is pushed onto stack.

iii) Accept: Announce successful completion of parsing.

iv) Error: Discover syntax error and call error recovery routine.

5. After each iteration, call check() function to find out if we can Reduce, otherwise keep shifting till input is $.
6. If only $S exists in input and stack, accept

the string otherwise reject the string.

7. Stop.

## MANUAL WORKING

Input Grammar : ① $E \rightarrow 2E2$
② $E \rightarrow 3E3$
③ $E \rightarrow 4$

Input string : 32423

Performing shift Reduce Parsing :

| Stack | Input | Action |
|---|---|---|
| $ | 32423$ | Shift |
| $3 | 2423$ | Shift |
| $32 | 423$ | Shift |
| $324 | 23$ | Reduce by ③ |
| $32E | 23$ | Shift |
| $32E2 | 3$ | Reduce by ① |
| $3E | 3$ | Shift |
| $3E3 | $ | Reduce by ② |
| $E | $ | ACCEPT. |

As only starting symbol was left in stack, input 32423 is accepted.

Sashrika Surya
RA1911027010092
Section: N2
Lab Batch: 1
Date: 11th March 2022

# Lab 7: Compiler Design
# Shift Reduce Parsing

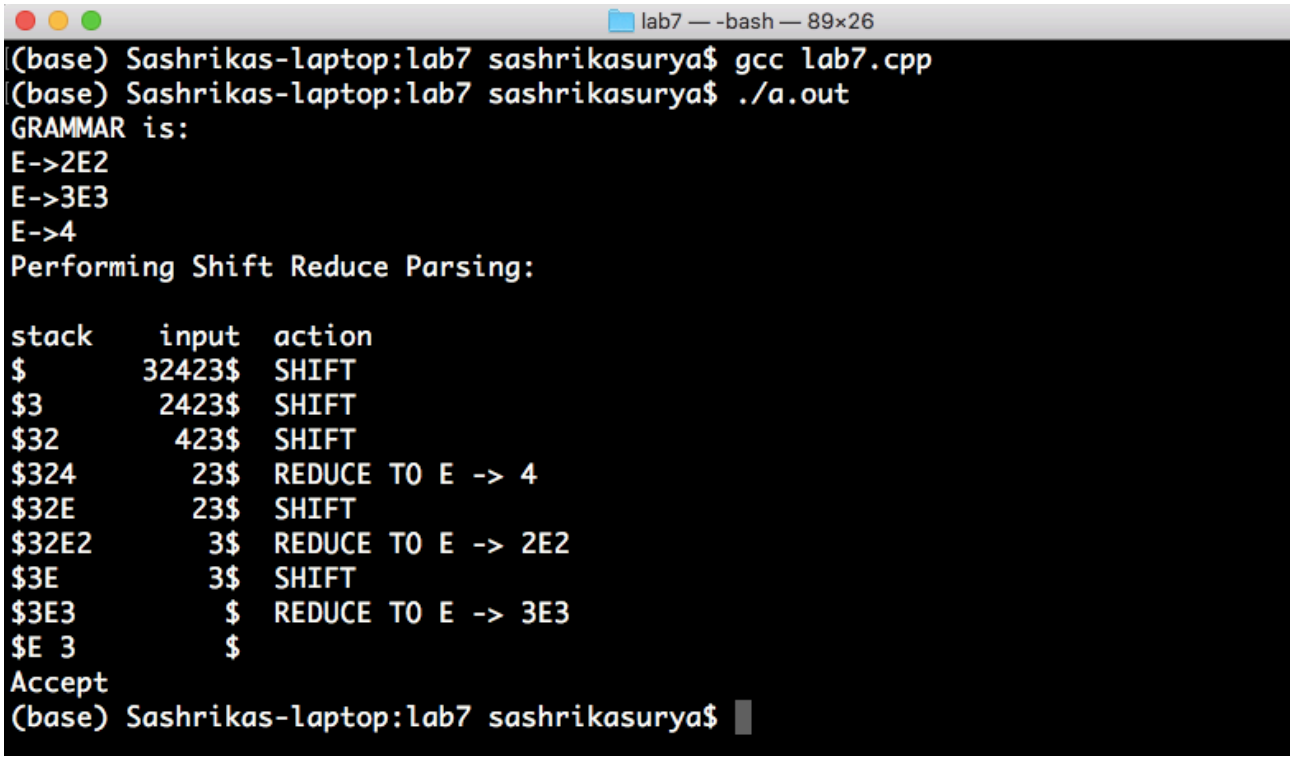
## Code

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>

int z = 0, i = 0, j = 0, c = 0;

char a[16], ac[20], stk[15], act[10];

// This Function will check whether
// the stack contain a production rule
// which is to be Reduce.
// Rules can be E->2E2 , E->3E3 , E->4
void check()
{
   // Coping string to be printed as action
   strcpy(ac,"REDUCE TO E -> ");

   // c=length of input string
   for(z = 0; z < c; z++)
   {
      //checking for producing rule E->4
      if(stk[z] == '4')
      {
         printf("%s4\n", ac);
         stk[z] = 'E';
         stk[z + 1] = ' ';

         //pinting action
         printf("$%s %s$ ", stk, a);
      }
   }
```

```
    for(z = 0; z < c - 2; z++)
    {
       //checking for another production
       if(stk[z] == '2' && stk[z + 1] == 'E' &&
                           stk[z + 2] == '2')
       {
          printf("%s2E2\n", ac);
          stk[z] = 'E';
          stk[z + 1] = ' ';
          stk[z + 2] = ' ';
          printf("$%s %s$ ", stk, a);
          i = i - 2;
       }

    }

    for(z=0; z<c-2; z++)
    {
       //checking for E->3E3
       if(stk[z] == '3' && stk[z + 1] == 'E' &&
                           stk[z + 2] == '3')
       {
          printf("%s3E3\n", ac);
          stk[z]='E';
          stk[z + 1]=' ';
          stk[z + 1]=' ';
          printf("$%s %s$ ", stk, a);
          i = i - 2;
       }
    }
    return ; //return to main
}

//Driver Function
int main()
{
    printf("GRAMMAR is:\nE->2E2 \nE->3E3 \nE->4");
    printf("\nPerforming Shift Reduce Parsing: \n");

    // a is input string
    strcpy(a,"32423");

    // strlen(a) will return the length of a to c
    c=strlen(a);
```

```
    // "SHIFT" is copied to act to be printed
    strcpy(act,"SHIFT");

    // This will print Lables (column name)
    printf("\nstack\t input \taction");

    // This will print the initial
    // values of stack and input
    printf("\n$ %s$ ", a);

    // This will Run upto length of input string
    for(i = 0; j < c; i++, j++)
    {
        // Printing action
        printf("%s\n", act);

        // Pushing into stack
        stk[i] = a[j];
        stk[i + 1] = ' ';

        // Moving the pointer
        a[j]=' ';

        // Printing action
        printf("$%s %s$ ", stk, a);

        // Call check function ..which will
        // check the stack whether its contain
        // any production or not
        check();
    }

    // Rechecking last time if contain
    // any valid production then it will
    // replace otherwise invalid
    check();

    // if top of the stack is E(starting symbol)
    // then it will accept the input
    if(stk[0] == 'E' && stk[1] == ' ')
        printf("\nAccept\n");
    else //else reject
        printf("\nReject\n");
}
```

**Output**




**Result:** The given objective was reached as shift reduce parsing was performed.