EXP 7

SHIFT REDUCE PARSING TABLE

AIM: To write a program for displaying the shift reduce parsing table of a given production

ALGORITHM:

1. Start the program.
2. Initialize the required variables.
3. Get the number of coordinates and productions from the user.
4. Perform the following

Shift: This involves moving symbols from the input buffer onto the stack.

Reduce: If the handle appears on top of the stack then, its reduction by using appropriate production rule is done i.e. RHS of a production rule is popped out of a stack and LHS of a production rule is pushed onto the stack.

Accept: If only the start symbol is present in the stack and the input buffer is empty then, the parsing action is called accept. When accepted action is obtained, it is means successful parsing is done.

Error: This is the situation in which the parser can neither perform shift action nor reduce action and not even accept action.

5. Print the resulting stack.
6. Print if the grammar is accepted or not.
7. Exit the program.

CODE:

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

//Global Variables
int z = 0, i = 0, j = 0, c = 0;

// Modify array size to increase
// length of string to be parsed
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()
{
   // Copying 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", ac);
         stk[z] = 'E';
         stk[z + 1] = '\0';
```

```
        //printing action
        printf("\n$%s\t%s$\t", 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", ac);
        stk[z] = 'E';
        stk[z + 1] = '\0';
        stk[z + 2] = '\0';
        printf("\n$%s\t%s$\t", 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", ac);
        stk[z]='E';
        stk[z + 1]='\0';
        stk[z + 1]='\0';
```

```
            printf("\n$%s\t%s$\t", stk, a);
            i = i - 2;
        }
    }
    return ; //return to main
}

//Driver Function
int main()
{
    printf("GRAMMAR is -\nE->2E2 \nE->3E3 \nE->4\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 Labels (column name)
    printf("\nstack \t input \t action");

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

    // This will Run upto length of input string
    for(i = 0; j < c; i++, j++)
    {
```

```
        // Printing action
        printf("%s", act);

        // Pushing into stack
        stk[i] = a[j];
        stk[i + 1] = '\0';

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

        // Printing action
        printf("\n$%s\t%s$\t", 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] == '\0')
        printf("Accept\n");
    else //else reject
        printf("Reject\n");
}
```

PRODUCTION:

Example 2 – Consider the grammar

E –> 2E2

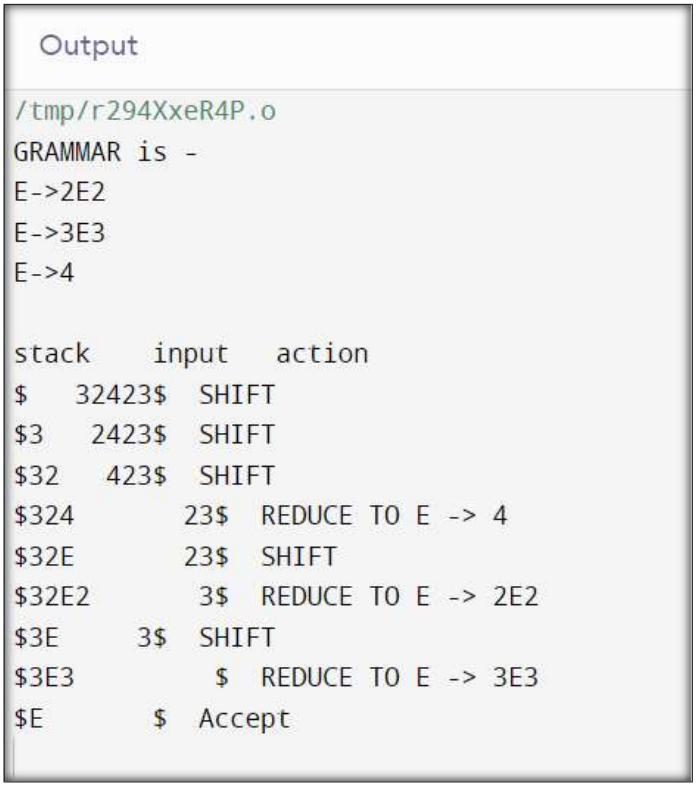
E –> 3E3

E –> 4

Perform Shift Reduce parsing for input string "32423".

Stack	Input Buffer	Parsing Action
$	32423$	Shift
$3	2423$	Shift
$32	423$	Shift
$324	23$	Reduce by E --> 4
$32E	23$	Shift
$32E2	3$	Reduce by E --> 2E2
$3E	3$	Shift
$3E3	$	Reduce by E --> 3E3
$E	$	Accept

OUTPUT:



```
Output

/tmp/r294XxeR4P.o
GRAMMAR is -
E->2E2
E->3E3
E->4

stack    input   action
$   32423$  SHIFT
$3   2423$  SHIFT
$32   423$  SHIFT
$324       23$  REDUCE TO E -> 4
$32E       23$  SHIFT
$32E2       3$  REDUCE TO E -> 2E2
$3E     3$  SHIFT
$3E3         $  REDUCE TO E -> 3E3
$E       $  Accept
```

RESULT:

The program for finding shift reduce parsing table is successfully compiled and executed.

RA1911003010387

CHITRALEKHA.CH