Name: Atandrit Chatterjee

Reg. No.: RA2011031010042

Experiment-7 Implement Shift Reduce Parsing

Aim:

To implement Shift Reduce Parsing in C/C++.

Procedure:

Shift Reduce parser attempts for the construction of parse in a similar manner as done in bottom up parsing i.e. the parse tree is constructed from leaves(bottom) to the root(up). A more general form of shift reduce parser is LR parser.
This parser requires some data structures i.e.

- A input buffer for storing the input string.
- A stack for storing and accessing the production rules.

Basic Operations –

- **Shift:** This involves moving of symbols from 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 production rule is popped out of stack and LHS of production rule is pushed onto the stack.
- **Accept:** If only start symbol is present in the stack and the input buffer is empty then, the parsing action is called accept. When accept 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.

Code:

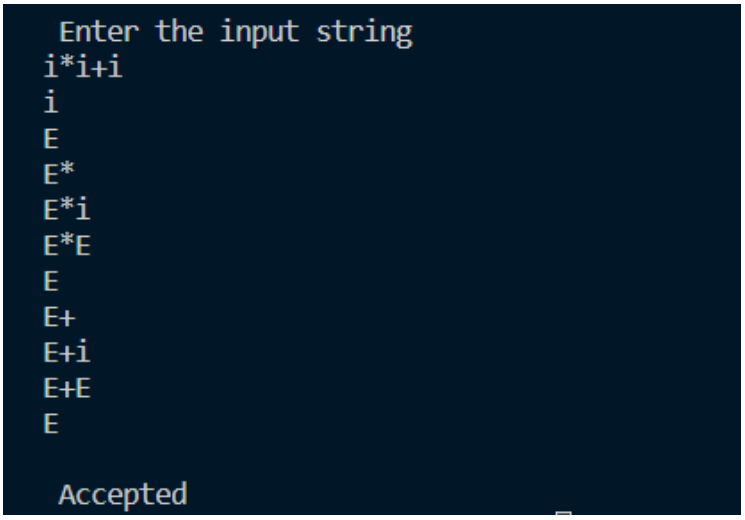
```
#include <bits/stdc++.h>
struct prodn {
   char p1[10];
   char p2[10];
};
int main() {
   system("cls");
   char input[20], stack[50], temp[50], ch[2], *t1, *t2, *t;
   int i, j, s1, s2, s, count = 0;
   struct prodn p[10];
   FILE *fp = fopen("sr_input.txt", "r");
   stack[0] = '\0';
   printf("\n Enter the input string\n");
   scanf("%s", &input);
   while (!feof(fp)) {
      fscanf(fp, "%s\n", temp);
      t1 = strtok(temp, "->");
      t2 = strtok(NULL, "->");
      strcpy(p[count].p1, t1);
      strcpy(p[count].p2, t2);
      count++;
   }
   i = 0;
   while (1) {
      if (i < strlen(input))
      {
         ch[0] = input[i];
         ch[1] = '\0';
         i++;
         strcat(stack, ch);
```

```
            printf("%s\n", stack);
        }
        for (j = 0; j < count; j++) {
            t = strstr(stack, p[j].p2);
            if (t != NULL)
            {
                s1 = strlen(stack);
                s2 = strlen(t);
                s = s1 - s2;
                stack[s] = '\0';
                strcat(stack, p[j].p1);
                printf("%s\n", stack);
                j = -1;
            }
        }
        if (strcmp(stack, "E") == 0 && i == strlen(input))
        {
            printf("\n Accepted\n");
            break;
        }
        if (i == strlen(input))
        {
            printf("\n Not Accepted\n");
            break;
        }
    }
    system("pause");
    return 0;
}
```

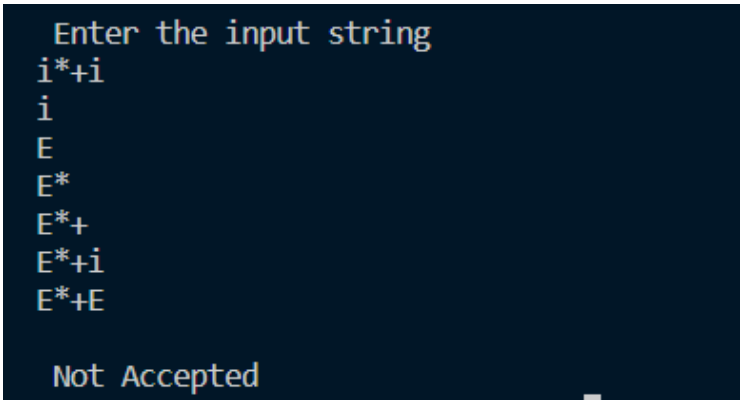
Input File:

```
E->E+E
E->E*E
E->i
```

Output 1:



Output 2:



Result:

The implementation of Shift Reduce Parsing was successful.