Exp. No. 14

Implement the concept of Shift reduce parsing in C Programming.

Program:

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

char ip_sym[15], stack[15];
int ip_ptr = 0, st_ptr = 0, len, i;
char temp[2], temp2[2];
char act[15];

void check();

int main() {
   printf("\n\t\tSHIFT REDUCE PARSER\n");
   printf("\nGRAMMAR\n");
   printf("\nE -> E+E\nE -> E/E\nE -> E*E\nE -> a/b\n");

   printf("\nEnter the input symbol: ");
   fgets(ip_sym, sizeof(ip_sym), stdin);
   ip_sym[strcspn(ip_sym, "\n")] = '\0'; // Remove newline character

   printf("\n\tStack Implementation Table");
   printf("\nStack\t\tInput Symbol\t\tAction");
   printf("\n----------------------------------------\n");

   printf("\n$\t\t%s$\t\t\t--", ip_sym);

   strcpy(act, "shift ");
   temp[0] = ip_sym[ip_ptr];
   temp[1] = '\0';
   strcat(act, temp);

   len = strlen(ip_sym);

   for (i = 0; i < len; i++) {
      stack[st_ptr] = ip_sym[ip_ptr];
      stack[st_ptr + 1] = '\0';
      ip_sym[ip_ptr] = ' ';
      ip_ptr++;
```

```
        printf("\n$%s\t\t%s$\t\t\t%s", stack, ip_sym, act);

        strcpy(act, "shift ");
        temp[0] = ip_sym[ip_ptr];
        temp[1] = '\0';
        strcat(act, temp);

        check();
        st_ptr++;
    }

    st_ptr++;
    check();

    return 0;
}

void check() {
    int flag = 0;
    temp2[0] = stack[st_ptr];
    temp2[1] = '\0';

    // Checking for terminals 'a' or 'b' and replacing them with 'E'
    if ((!strcmp(temp2, "a")) || (!strcmp(temp2, "b"))) {
        stack[st_ptr] = 'E';
        stack[st_ptr + 1] = '\0';
        printf("\n$%s\t\t%s$\t\t\tE->%s", stack, ip_sym, temp2);
        flag = 1;
    }

    // Checking for operators: '+', '*', or '/'
    if ((!strcmp(temp2, "+")) || (!strcmp(temp2, "*")) || (!strcmp(temp2,
"/"))) {
        flag = 1;
    }

    // Checking for expressions that can be reduced to E
    if ((!strcmp(stack, "E+E")) || (!strcmp(stack, "E/E")) || (!strcmp(stack,
"E*E"))) {
        strcpy(stack, "E");
        st_ptr = 0;

            printf("\n$%s\t\t%s$\t\t\tE->E+E", stack, ip_sym);
```

```
        else if (!strcmp(stack, "E/E"))
            printf("\n$%s\t\t%s$\t\t\tE->E/E", stack, ip_sym);
        else if (!strcmp(stack, "E*E"))
            printf("\n$%s\t\t%s$\t\t\tE->E*E", stack, ip_sym);

        flag = 1;
    }

    // Accept condition
    if (!strcmp(stack, "E") && ip_ptr == len) {
        printf("\n$%s\t\t%s$\t\t\tACCEPT", stack, ip_sym);
        exit(0);
    }

    // Reject condition
    if (flag == 0) {
        printf("\n$%s\t\t%s$\t\t\tREJECT", stack, ip_sym);
        exit(0);
    }
}
```

Output

```
        SHIFT REDUCE PARSER

GRAMMAR

E -> E+E
E -> E/E
E -> E*E
E -> a/b

Enter the input symbol: a+b

    Stack Implementation Table
Stack       Input Symbol        Action
-----------------------------------------

$       a+b$            --
$a       +b$            shift a
$E       +b$            E->a
$E+       b$            shift +
$E+b           $            shift b
$E+E           $            E->b
$E         $            ACCEPT

=== Code Execution Successful ===
```