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

#define MAX_RULE_COUNT 10
#define MAX_RULE_LENGTH 10
#define MAX_STRING_LENGTH 20

int main()
{
        char input[MAX_STRING_LENGTH], stack[MAX_STRING_LENGTH], temp[MAX_STRING_LENGTH], ch[2], *token1, *token2, *substring;
        int i, j, stacklength, substringlength, stacktop, prodrulecount = 0;
        char left[MAX_RULE_COUNT][MAX_RULE_LENGTH];
        char right[MAX_RULE_COUNT][MAX_RULE_LENGTH];

        stack[0] = '\0';

        // User input for the number of production rules
        printf("\n Kindly enter the number of production rules: ");
        scanf("%d", &prodrulecount);

        printf("\n");
        // User input for each production rule in the form 'LHS->RHS'
        for (i = 0; i < prodrulecount; i++)
        {
         printf(" Kindly enter Production Rule %d: ",i+1);
         scanf("%s", temp);
         token1 = strtok(temp, "->");
         token2 = strtok(NULL, "->");
         strcpy(left[i], token1);
         strcpy(right[i], token2);
        }

        // User input for the input string
        printf("\n Kindly enter the input string: ");
        scanf("%s", input);

        printf(" Stack    Input   Action\n");
        printf(" --------------------------------\n");
        i = 0;

        while (1)
        {
         //Shift Operation
         if (i < strlen(input))  // If there are more characters in the input string, add the next character to the stack
         {
             ch[0] = input[i];
             ch[1] = '\0';
             i++;
             strcat(stack, ch);
             printf(" %s\t", stack);
             for (int k = i; k < strlen(input); k++)
             {
                  printf("%c", input[k]);
             }
             printf("\tShift %s\n", ch);
         }

        //Reduce Operation
         for (j = 0; j < prodrulecount; j++)    // Iterate through the production rules
         {
             // Check if the right-hand side of the production rule matches a substring in the stack
             substring = strstr(stack, right[j]);
             if (substring != NULL)             //if match found
             {
                  // Replace the matched substring with the left-hand side of the production rule
                  stacklength = strlen(stack);
                  substringlength = strlen(substring);
                  stacktop = stacklength - substringlength;
                  stack[stacktop] = '\0';
                  strcat(stack, left[j]);
                  printf(" %s\t", stack);
                  for (int k = i; k < strlen(input); k++)
                  {
                      printf("%c", input[k]);
                  }
                  printf("\tReduce %s->%s\n", left[j], right[j]);
                  j = -1; // Restart the loop to ensure immediate reduction of the newly derived production rule
             }
         }

         // Check if the stack contains only the start symbol and if the entire input string has been processed
         if (strcmp(stack, left[0]) == 0 && i == strlen(input))
         {
             printf("\n Input string accepted\n");
             break;
         }

         // Check if the entire input string has been processed but the stack doesn't match the start symbol
         if (i == strlen(input))
         {
             printf("\n Input string rejected\n");
             break;
         }
        }

        return 0;
}
```