LAB

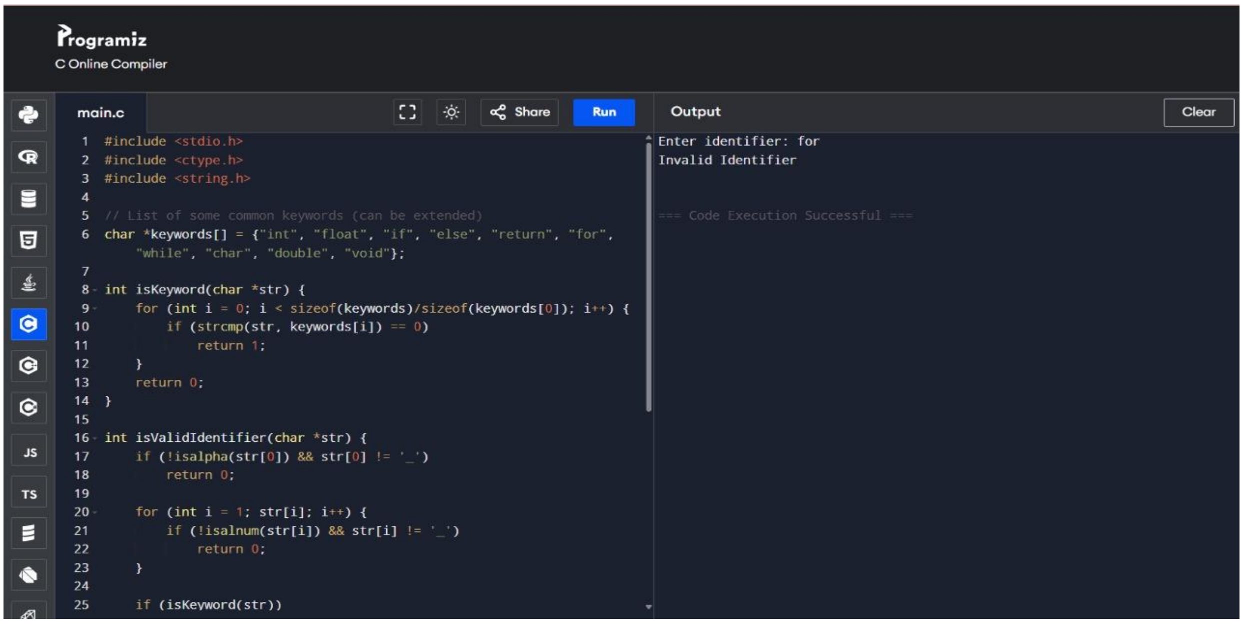
NAME : ASWANTH N N

REG NO : 192424123

COURSE : CSA1423-Compiler Design

EXP 3

Develop a lexical Analyzer to test whether a given identifier is valid or not.

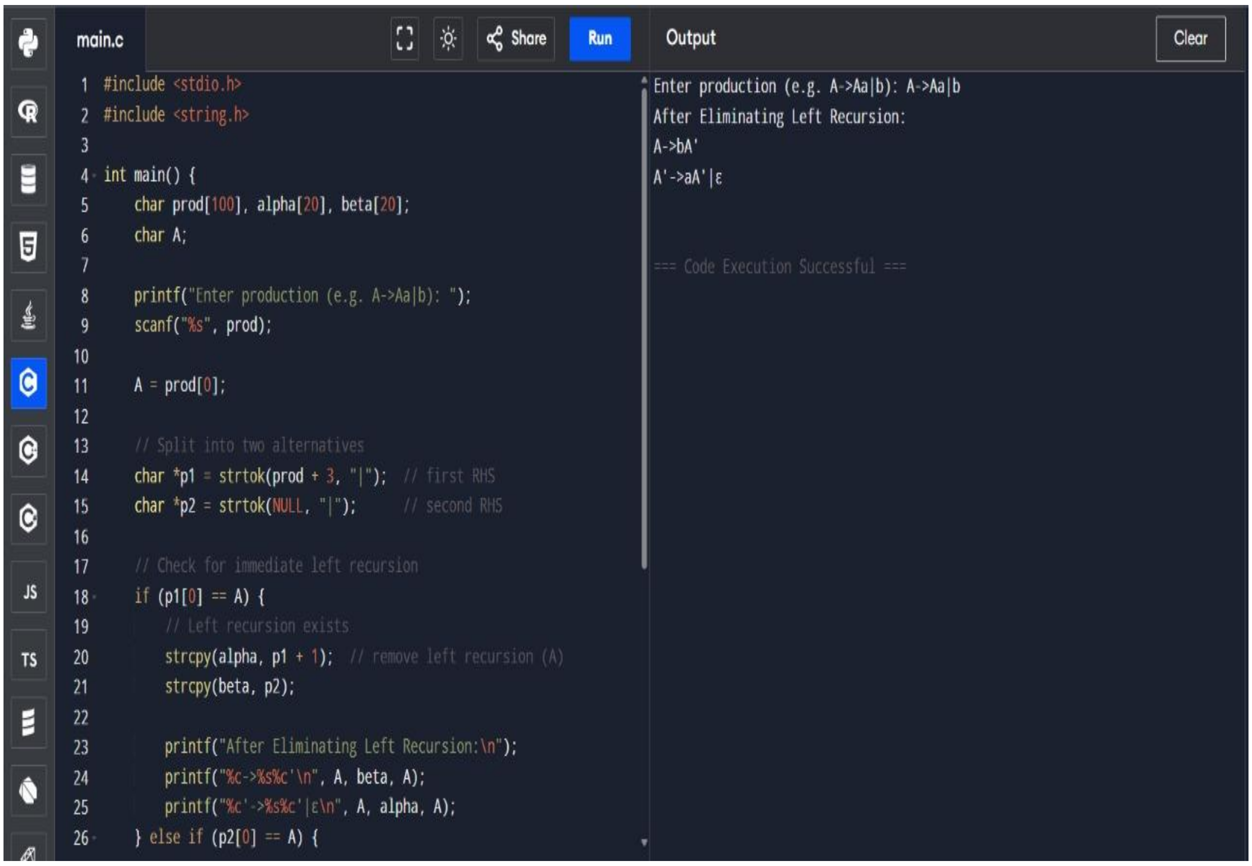


```c
#include <stdio.h>
#include <ctype.h>
#include <string.h>

// List of some common keywords (can be extended)
char *keywords[] = {"int", "float", "if", "else", "return", "for",
    "while", "char", "double", "void"};

int isKeyword(char *str) {
    for (int i = 0; i < sizeof(keywords)/sizeof(keywords[0]); i++) {
        if (strcmp(str, keywords[i]) == 0)
            return 1;
    }
    return 0;
}

int isValidIdentifier(char *str) {
    if (!isalpha(str[0]) && str[0] != '_')
        return 0;

    for (int i = 1; str[i]; i++) {
        if (!isalnum(str[i]) && str[i] != '_')
            return 0;
    }

    if (isKeyword(str))
```

```
Enter identifier: for
Invalid Identifier


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

EXP 4

Implement a C program to eliminate left recursion.

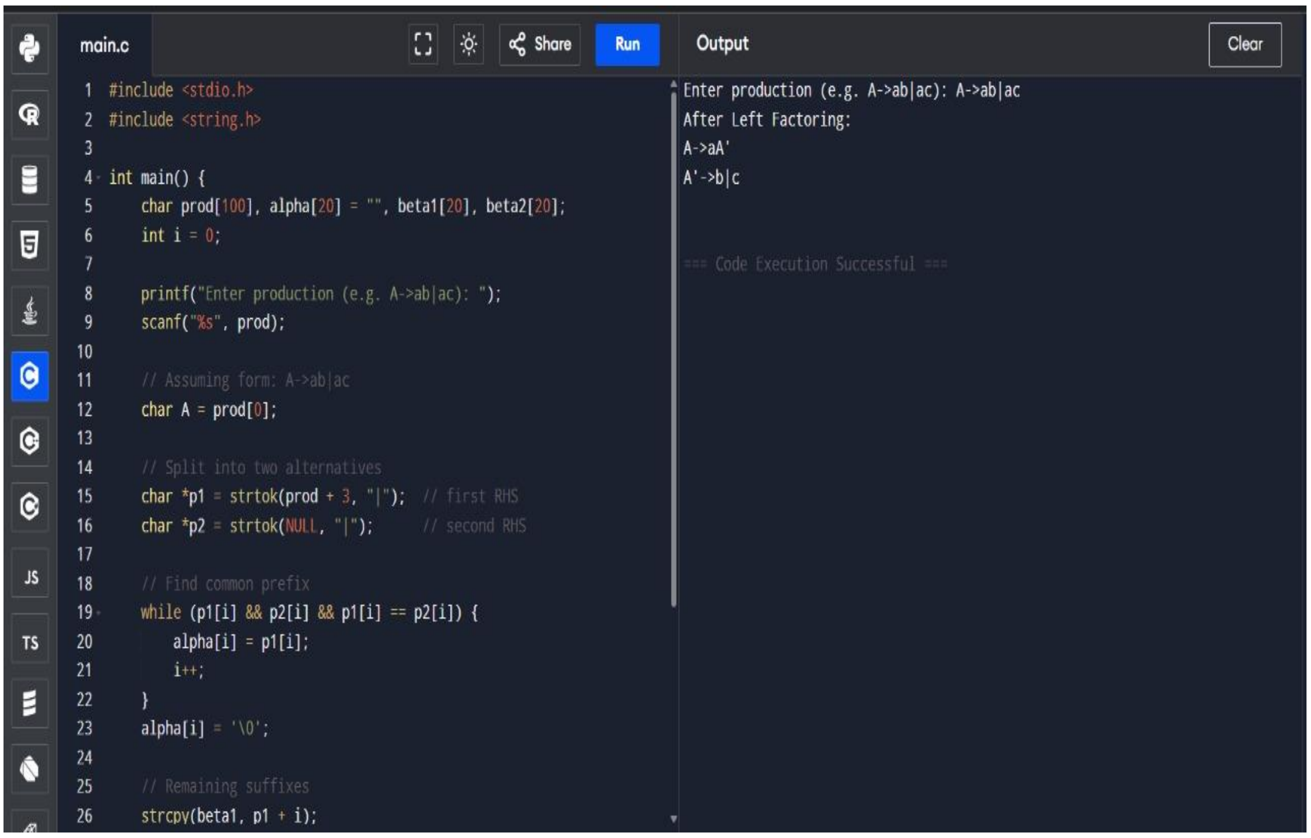


```c
#include <stdio.h>
#include <string.h>

int main() {
    char prod[100], alpha[20], beta[20];
    char A;

    printf("Enter production (e.g. A->Aa|b): ");
    scanf("%s", prod);

    A = prod[0];

    // Split into two alternatives
    char *p1 = strtok(prod + 3, "|");  // first RHS
    char *p2 = strtok(NULL, "|");      // second RHS

    // Check for immediate left recursion
    if (p1[0] == A) {
        // Left recursion exists
        strcpy(alpha, p1 + 1);  // remove left recursion (A)
        strcpy(beta, p2);

        printf("After Eliminating Left Recursion:\n");
        printf("%c->%s%c'\n", A, beta, A);
        printf("%c'->%s%c'|ε\n", A, alpha, A);
    } else if (p2[0] == A) {
```

```
Enter production (e.g. A->Aa|b): A->Aa|b
After Eliminating Left Recursion:
A->bA'
A'->aA'|ε


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

EXP 5

Implement a C program to eliminate left factoring.



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

int main() {
    char prod[100], alpha[20] = "", beta1[20], beta2[20];
    int i = 0;

    printf("Enter production (e.g. A->ab|ac): ");
    scanf("%s", prod);

    // Assuming form: A->ab|ac
    char A = prod[0];

    // Split into two alternatives
    char *p1 = strtok(prod + 3, "|");  // first RHS
    char *p2 = strtok(NULL, "|");      // second RHS

    // Find common prefix
    while (p1[i] && p2[i] && p1[i] == p2[i]) {
        alpha[i] = p1[i];
        i++;
    }
    alpha[i] = '\0';

    // Remaining suffixes
    strcpy(beta1, p1 + i);
```

```
Enter production (e.g. A->ab|ac): A->ab|ac
After Left Factoring:
A->aA'
A'->b|c


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