6.    Implement a C program to eliminate left recursion.

#include <stdio.h>

#include <string.h>

int main() {

char input[100], alpha[10], beta[10];

char nonTerminal;

printf("Enter production (Example: A->Aa|b): ");

scanf("%s", input);

nonTerminal = input[0];

if (input[3] == nonTerminal) {

// Left recursive

int i = 4, j = 0;

while (input[i] != '|') {

alpha[j++] = input[i++];

}

alpha[j] = '\0';

i++; // Skip '|'

j = 0;

while (input[i] != '\0') {

beta[j++] = input[i++];

}

beta[j] = '\0';

printf("Left Recursion Detected.\n");

printf("After eliminating left recursion:\n")

printf("%c -> %s%c'\n", nonTerminal, beta, nonTerminal);

printf("%c' -> %s%c' | ε\n", nonTerminal, alpha, nonTerminal);

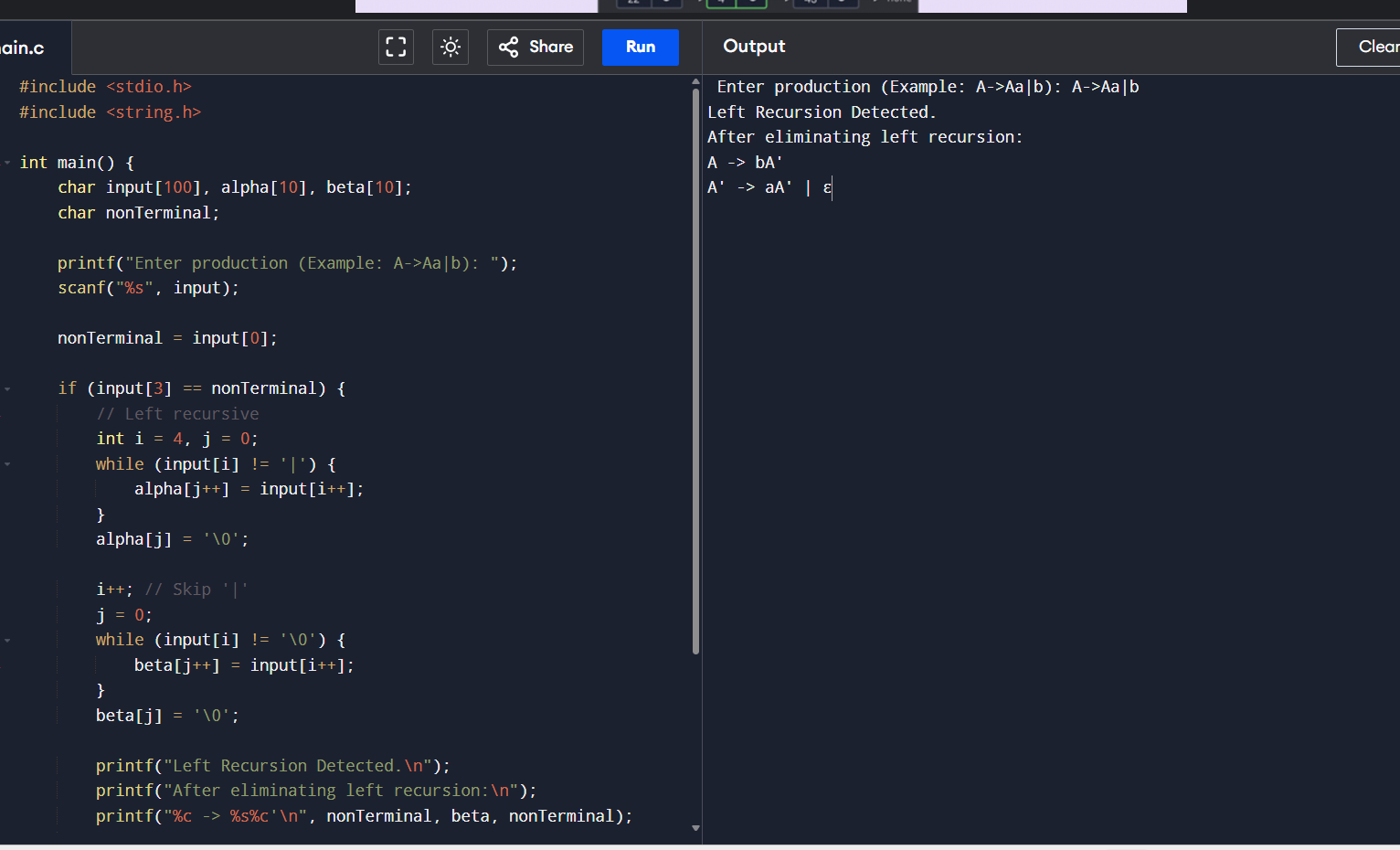
} else {

printf("No Left Recursion in the given production.\n");

}

return 0;

}



7.    Implement a C program to eliminate left factoring.

#include <stdio.h>

#include <string.h>

int main() {

char prod[100], common[10], part1[10], part2[10];

char nonTerminal;

printf("Enter production (Example: A->abc|abd): ");

scanf("%s", prod);

nonTerminal = prod[0];

int i = 3, j = 0;

while (prod[i] != '|') {

part1[j++] = prod[i++];

}

part1[j] = '\0';

i++; j = 0;

while (prod[i] != '\0') {

part2[j++] = prod[i++];

}

part2[j] = '\0';

// Find common prefix

int k = 0;

while (part1[k] != '\0' && part2[k] != '\0' && part1[k] == part2[k]) {

common[k] = part1[k];

k++;

}

common[k] = '\0';

if (strlen(common) == 0) {

printf("No common prefix. Left factoring not required.\n");

} else {

printf("Left Factoring Applied.\n");

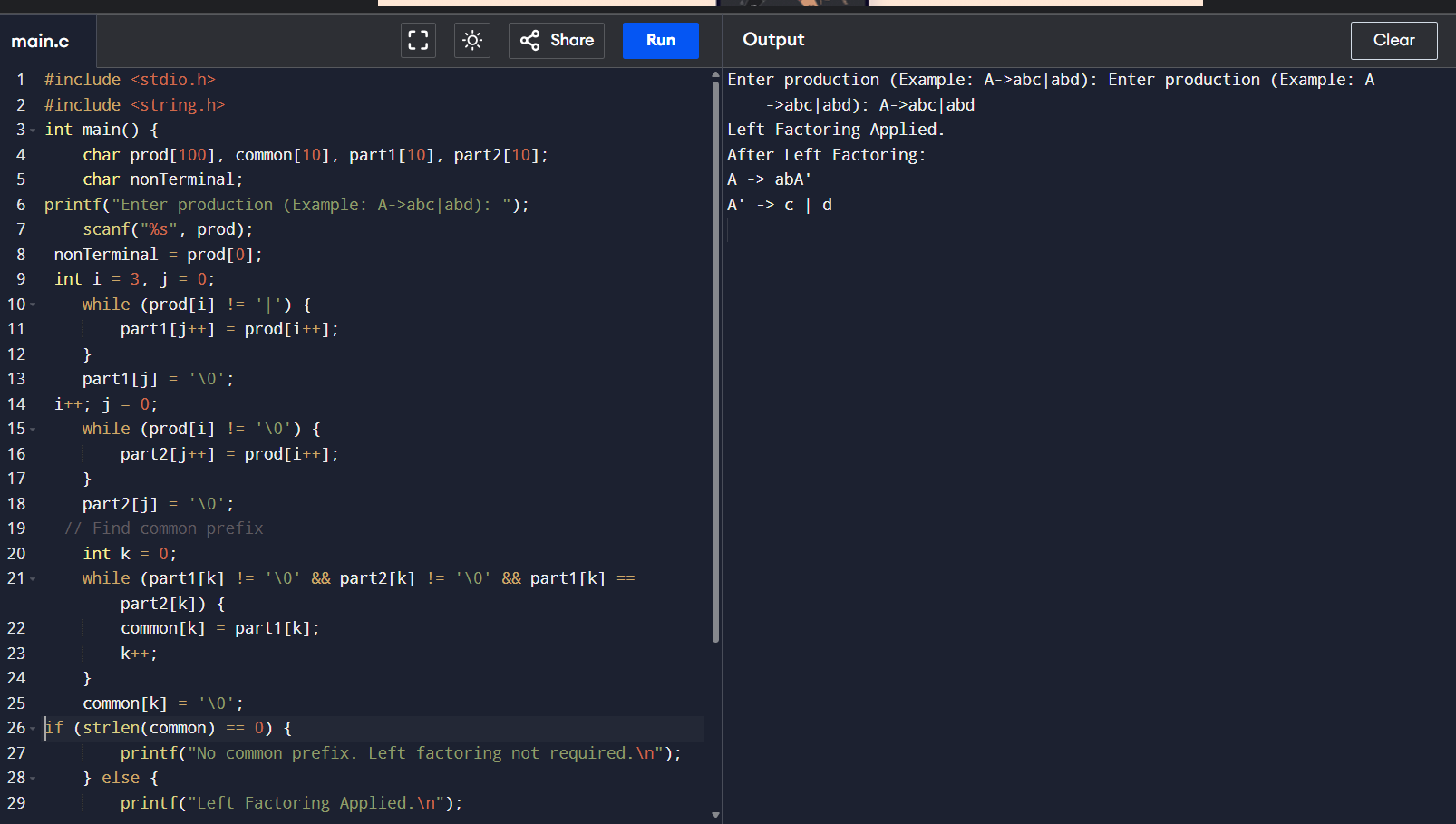
printf("After Left Factoring:\n");

printf("%c -> %s%c'\n", nonTerminal, common, nonTerminal);

printf("%c' -> %s | %s\n", nonTerminal, &part1[k], &part2[k]);

}

return 0;

}

8.  Implement a C program to perform symbol table operations.

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#define SIZE 100

struct Symbol {

char name[20];

char type[10];

};

struct Symbol table[SIZE];

int count = 0;

void insert() {

char name[20], type[10];

printf("Enter identifier name: ");

scanf("%s", name);

// Check if already exists

for (int i = 0; i < count; i++) {

if (strcmp(table[i].name, name) == 0) {

printf("Identifier already exists.\n");

return;

}

}

printf("Enter type: ");

scanf("%s", type);

strcpy(table[count].name, name);

strcpy(table[count].type, type);

count++;

printf("Inserted Successfully.\n");

}

void display() {

printf("\nSymbol Table:\n");

printf("------------------------\n");

printf("Name\t\tType\n");

printf("------------------------\n");

for (int i = 0; i < count; i++) {

printf("%s\t\t%s\n", table[i].name, table[i].type);

}

}

void search() {

char name[20];

printf("Enter identifier to search: ");

scanf("%s", name);

for (int i = 0; i < count; i++) {

if (strcmp(table[i].name, name) == 0) {

printf("Identifier found at position %d\n", i + 1);

return;

}

}

printf("Identifier not found.\n");

}

int main()

int choice;

while (1) {

printf("\n1. Insert\n2. Display\n3. Search\n4. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1: insert(); break;

case 2: display(); break;

case 3: search(); break;

case 4: exit(0);

default: printf("Invalid choice.\n");

}

}

return 0;

}