192311044

Girija B

18.02.2025

CSA1428 - Compiler Design

LAB ACTIVITY-2

1. Implement a C program to eliminate left recursion.

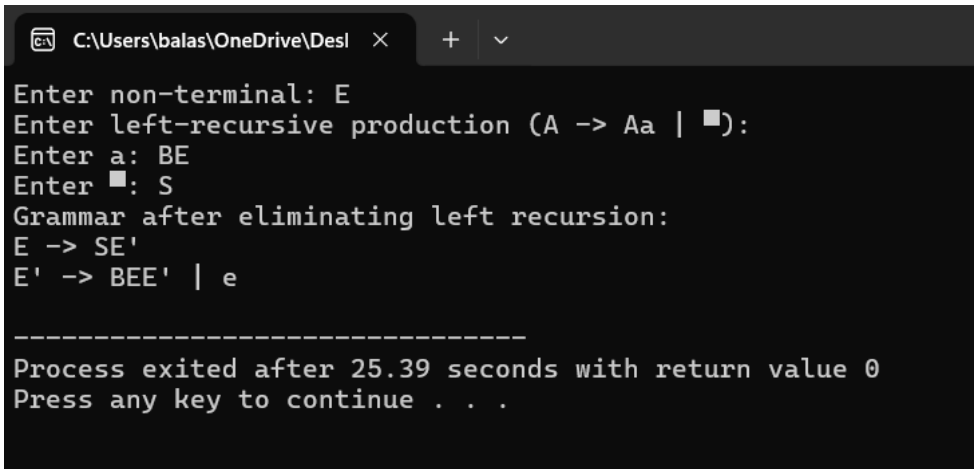
Code:

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

void eliminateLeftRecursion(char nonTerminal, char alpha[], char beta[]) {
    printf("Grammar after eliminating left recursion:\n");
    printf("%c -> %s%c'\n", nonTerminal, beta, nonTerminal);
    printf("%c' -> %s%c' | ε\n", nonTerminal, alpha, nonTerminal);
}

int main() {
    char nonTerminal, alpha[10], beta[10];
    printf("Enter non-terminal: ");
    scanf(" %c", &nonTerminal);
    printf("Enter left-recursive production (A -> Aα | β):\n");
    printf("Enter α: ");
    scanf("%s", alpha);
    printf("Enter β: ");
    scanf("%s", beta);
    eliminateLeftRecursion(nonTerminal, alpha, beta);
    return 0;
}
```

Output:



2. Implement a C program to eliminate left factoring.

Code:

```
#include <stdio.h>

#include <string.h>


void eliminateLeftFactoring(char nonTerminal, char common[], char alpha[], char beta[]) {

    printf("Grammar after eliminating left factoring:\n");

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

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

}


int main() {

    char nonTerminal, common[10], alpha[10], beta[10];

    printf("Enter non-terminal: ");

    scanf(" %c", &nonTerminal);

    printf("Enter common prefix: ");

    scanf("%s", common);

    printf("Enter α (remaining part of first production): ");

    scanf("%s", alpha);

    printf("Enter β (remaining part of second production): ");

    scanf("%s", beta);
```

```
    eliminateLeftFactoring(nonTerminal, common, alpha, beta);
    return 0;
}
```

Output:

```
C:\Users\balas\OneDrive\Desl

Enter non-terminal: A
Enter common prefix: ab
Enter a (remaining part of first production): S
Enter ■ (remaining part of second production): id
Grammar after eliminating left factoring:
A -> abA'
A' -> S | id | e

--------------------------------
Process exited after 24.39 seconds with return value 0
Press any key to continue . . .
```

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

Code:

```
#include <stdio.h>
#include <string.h>
#define SIZE 10
struct Symbol {
    char name[20];
    char type[10];
} table[SIZE];
int count = 0;
void insert() {
    if (count < SIZE) {
        printf("Enter symbol name: ");
        scanf("%s", table[count].name);
        printf("Enter type: ");
        scanf("%s", table[count].type);
        count++;
```

```
        printf("Symbol inserted successfully!\n");
    } else {
        printf("Symbol table full!\n");
    }
}
void search() {
    char name[20];
    printf("Enter symbol name to search: ");
    scanf("%s", name);

    for (int i = 0; i < count; i++) {
        if (strcmp(table[i].name, name) == 0) {
            printf("Symbol found: %s, Type: %s\n", table[i].name, table[i].type);
            return;
        }
    }
    printf("Symbol not found!\n");
}
void display() {
    printf("\nSymbol Table:\n");
    printf("Name\tType\n");
    for (int i = 0; i < count; i++) {
        printf("%s\t%s\n", table[i].name, table[i].type);
    }
}
int main() {
    int choice;
    while (1) {
        printf("\n1. Insert\n2. Search\n3. Display\n4. Exit\nEnter choice: ");
        scanf("%d", &choice);
        switch (choice) {
            case 1: insert(); break;
```

```
        case 2: search(); break;
        case 3: display(); break;
        case 4: return 0;
        default: printf("Invalid choice!\n");
    }
  }
}
```

Output:

```
1. Insert
2. Search
3. Display
4. Exit
Enter choice: 1
Enter symbol name: y
Enter type: int
Symbol inserted successfully!

1. Insert
2. Search
3. Display
4. Exit
Enter choice: 2
Enter symbol name to search: x
Symbol found: x, Type: int

1. Insert
2. Search
3. Display
4. Exit
Enter choice: 3

Symbol Table:
Name    Type
x       int
y       int
```

4. All languages have Grammar. When people frame a sentence we usually say whether the sentence is framed as per the rules of the Grammar or Not. Similarly use the same ideology , implement to check whether the given input string is satisfying the grammar or not .

Code:

(Sample Grammar: S -> aSb | ab)

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
int checkGrammar(char str[], int left, int right) {
  if (left > right) return 0;
  if (left == right - 1 && str[left] == 'a' && str[right] == 'b')
    return 1;
```

```
    if (str[left] == 'a' && str[right] == 'b')
        return checkGrammar(str, left + 1, right - 1);
    return 0;
}
int main() {
    char str[100];
    printf("Enter the input string: ");
    scanf("%s", str);
    int len = strlen(str);
    if (checkGrammar(str, 0, len - 1))
        printf("The string satisfies the grammar.\n");
    else
        printf("The string does not satisfy the grammar.\n");
    return 0;
}
```

Output:

```
C:\Users\balas\OneDrive\Desl

Enter the input string: ab
The string satisfies the grammar.

--------------------------------
Process exited after 14.63 seconds with return value 0
Press any key to continue . . .
```

```
C:\Users\balas\OneDrive\Desl

Enter the input string: abbbbb
The string does not satisfy the grammar.

--------------------------------
Process exited after 3.648 seconds with return value 0
Press any key to continue . . .
```

5. Write a C program to construct recursive descent parsing.

Code:

{For production ex:

E → T E'

E' → + T E' | ε

T → F T'

T' → * F T' | ε

F → (E) | id}

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
char input[100];
int index = 0;
void E();
void E_prime();
void T();
void T_prime();
void F();

void error() {
    printf("Error in parsing!\n");
    exit(0);
}

void match(char expected) {
    if (input[index] == expected)
        index++;
    else
        error();
}
```

```
void E() {
    T();
    E_prime();
}
void E_prime() {
    if (input[index] == '+') {
        match('+');
        T();
        E_prime();
    }
}
void T() {
    F();
    T_prime();
}
void T_prime() {
    if (input[index] == '*') {
        match('*');
        F();
        T_prime();
    }
}

void F() {
    if (input[index] == '(') {
        match('(');
        E();
        match(')');
    } else if (isalnum(input[index])) {
        match(input[index]);
    } else {
        error();
```

```
    }
}
int main() {
    printf("Enter an expression: ");
    scanf("%s", input);
    E();
    if (input[index] == '\0')
        printf("Parsing successful!\n");
    else
        printf("Parsing failed!\n");
    return 0;
}
```

Output:

```
C:\Users\balas\OneDrive\Desl

Enter an expression: a+b*c
Parsing successful!

--------------------------------
Process exited after 25.75 seconds with return value 0
Press any key to continue . . .
```

```
C:\Users\balas\OneDrive\Desl

Enter an expression: a+*
Error in parsing!

--------------------------------
Process exited after 12.42 seconds with return value 0
Press any key to continue . . .
```