Green University of Bangladesh

Department of Computer Science and Engineering (CSE)

Faculty of Sciences and Engineering

Semester: (Spring, Year: 2021), B.Sc. in CSE (Day)

LAB REPORT NO: 03

Course Title: Compiler

Course Code: CSE-306 **Section:** 193-DB

Lab Experiment Name: Write a C program to calculate FIRST of a regular expression**.**

Student Details

	Name	ID
1.	**Jakirul Islam**	**193002101**

Lab Date: 15.12.2021

Course Teacher's Name: Md. Atikuzzaman

[For Teachers use only: Don't Write Anything inside this box]

Lab Report Status

Marks: …… **Signature:.....**

Comments:..... **Date:.....**

LAB REPORT TEMPLATE

1. TITLE OF THE LAB EXPERIMENT

Write a C program to calculate FIRST of a regular expression

2. OBJECTIVES/AIM

In this program, we will calculate FIRST of a regular expression. The objective of this lab experiment is to find the grammar.

3. PROCEDURE / ANALYSIS / DESIGN

To compute FIRST(X) for all grammar symbols x, apply the following rules until no more terminals can be added to any FIRST set.

1. if X is terminal, then FIRST(X) is {X}.

2. if X is nonterminal and X-> aα is a production, then add a to FIRST(X). if X->€ to FIRST(X)

3. if -> Y1,Y2,…….Yk is a production, then for all i such that all of Y1,….Yi-1 are nonterminals and FIRST(Yj) contains € for j=1,2,…… i-1, add every non-€ symbol in FIRST(Y1) to FIRST(x). if V is in FIRST(Yj) for j=1,2,……k, then add € to FIRST(X).

4. IMPLEMENTATION

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


void main() {
 char t[5], nt[10], p[5][5], first[5][5], temp;
 int i, j, not, nont, k = 0, f = 0;
 printf("\nEnter the no. of Non-terminals in the grammer:");
 scanf("%d", & nont);
 printf("\nEnter the Non-terminals in the grammer:\n");
 for (i = 0; i < nont; i++) {
   scanf("\n%c", & nt[i]);
 }
 printf("\nEnter the no. of Terminals in the grammer: ( Enter e for absiline ) ");
 scanf("%d", & not);
 printf("\nEnter the Terminals in the grammer:\n");
 for (i = 0; i < not || t[i] == '$'; i++) {
   scanf("\n%c", & t[i]);
 }
```

```
for (i = 0; i < nont; i++) {
  p[i][0] = nt[i];
  first[i][0] = nt[i];
}
printf("\nEnter the productions :\n");
for (i = 0; i < nont; i++) {
  scanf("%c", & temp);
   printf("\nEnter the production for %c ( End the production with '$' sign ):", p[i][0]);
  for (j = 0; p[i][j] != '$';) {
    j += 1;
```

```
    scanf("%c", & p[i][j]);
  }
}
for (i = 0; i < nont; i++) {
  printf("\nThe production for %c -> ", p[i][0]);
  for (j = 1; p[i][j] != '$'; j++) {
    printf("%c", p[i][j]);
  }
}
for (i = 0; i < nont; i++) {
  f = 0;
  for (j = 1; p[i][j] != '$'; j++) {
    for (k = 0; k < not; k++) {
      if (f == 1)
        break;
      if (p[i][j] == t[k]) {
        first[i][j] = t[k];
        first[i][j + 1] = '$';
        f = 1;
        break;
      } else if (p[i][j] == nt[k]) {
        first[i][j] = first[k][j];
        if (first[i][j] == 'e')
          continue;
        first[i][j + 1] = '$';
        f = 1;
        break;
      }
    }
  }
}
 for (i = 0; i < nont; i++) {
   printf("\n\nThe first of %c -> ", first[i][0]);
```

```
   for (j = 1; first[i][j] != '$'; j++) {
     printf("%c\t", first[i][j]);
   }
 }
 getch();
}
```

5. TEST RESULT / OUTPUT

```
rudra:~/ $ ./fregex

Enter the no. of Non-terminals in the grammer: 3

Enter the Non-terminals in the grammer:
ERT

Enter the no. of Terminals in the grammer: ( Enter e for absiline ) 5

Enter the Terminals in the grammer:
ase*+

Enter the productions:

Enter the production for E ( End the production with '$' sign ): a+s$

Enter the production for R ( End the production with '$' sign ): e$

Enter the production for T ( End the production with '$' sign ): Rs$

The production for E -> a+s
The production for R -> e
The production for T -> Rs

The first of E -> a

The first of R -> e

The first of T -> e     s       s
rudra:~/ $
```

ANALYSIS AND DISCUSSION

In this lab, I learned how to calculate FIRST of a regular expression and learned how to write code for calculating the FIRST.