Name: Atandrit Chatterjee

Reg. No.: RA2011031010042

Experiment-2 Conversion from Regular Expression to NFA

Aim:

To write a program for converting Regular Expression to NFA.

Procedure:

1. Start
2. Get the input from the user
3. Initialize separate variables and functions for Postfix, Display and NFA
4. Create separate methods for different operators like +, *,.
5. By using Switch case Initialize different cases for the input
6. For ‘. ' Operator Initialize a separate method by using various stack functions do the same for the other operators like ' * ' and ' + '.
7. Regular expression is in the form like a.b (or) a+b
8. Display the output
9. Stop

Code:

```
#include <bits/stdc++.h>
int main()
{
   system("cls");
   char reg[20];
```

```
int q[20][3], i, j, len, a, b;
for (a = 0; a < 20; a++)
{
    for (b = 0; b < 3; b++)
    {
        q[a][b] = 0;
    }
}
printf("Regular expression: \n");
scanf("%s", reg);
len = strlen(reg);
i = 0;
j = 1;
while (i < len)
{
    if (reg[i] == 'a' && reg[i + 1] != '/' && reg[i + 1] != '*')
    {
        q[j][0] = j + 1;
        j++;
    }
    if (reg[i] == 'b' && reg[i + 1] != '/' && reg[i + 1] != '*')
    {
        q[j][1] = j + 1;
        j++;
    }
    if (reg[i] == 'e' && reg[i + 1] != '/' && reg[i + 1] != '*')
    {
        q[j][2] = j + 1;
        j++;
    }
```

```
if (reg[i] == 'a' && reg[i + 1] == '/' && reg[i + 2] == 'b')
{
    q[j][2] = ((j + 1) * 10) + (j + 3);
    j++;
    q[j][0] = j + 1;
    j++;
    q[j][2] = j + 3;
    j++;
    q[j][1] = j + 1;
    j++;
    q[j][2] = j + 1;
    j++;
    i = i + 2;
}
if (reg[i] == 'b' && reg[i + 1] == '/' && reg[i + 2] == 'a')
{
    q[j][2] = ((j + 1) * 10) + (j + 3);
    j++;
    q[j][1] = j + 1;
    j++;
    q[j][2] = j + 3;
    j++;
    q[j][0] = j + 1;
    j++;
    q[j][2] = j + 1;
    j++;
    i = i + 2;
}
if (reg[i] == 'a' && reg[i + 1] == '*')
{
```

```
        q[j][2] = ((j + 1) * 10) + (j + 3);
        j++;
        q[j][0] = j + 1;
        j++;
        q[j][2] = ((j + 1) * 10) + (j - 1);
        j++;
    }
    if (reg[i] == 'b' && reg[i + 1] == '*')
    {
        q[j][2] = ((j + 1) * 10) + (j + 3);
        j++;
        q[j][1] = j + 1;
        j++;
        q[j][2] = ((j + 1) * 10) + (j - 1);
        j++;
    }
    if (reg[i] == ')' && reg[i + 1] == '*')
    {
        q[0][2] = ((j + 1) * 10) + 1;
        q[j][2] = ((j + 1) * 10) + 1;
        j++;
    }
    i++;
}
printf("\nTransition function\n");
for (i = 0; i <= j; i++)
{
    if (q[i][0] != 0)
        printf("\n q[%d,a]-->%d", i, q[i][0]);
    if (q[i][1] != 0)
```

```
            printf("\n q[%d,b]-->%d", i, q[i][1]);
        if (q[i][2] != 0)
        {
            if (q[i][2] < 10)
                printf("\n q[%d,e]-->%d", i, q[i][2]);
            else
                printf("\n q[%d,e]-->%d & %d", i, q[i][2] / 10, q[i][2] % 10);
        }
    }
    printf("\n");
    system("pause");
    return 0;
}
```

Output:

```
Regular expression:
(a|b)*abb

Transition function

 q[0,e]-->4 & 1
 q[1,a]-->2
 q[2,b]-->3
 q[3,e]-->4 & 1
 q[4,a]-->5
 q[5,b]-->6
 q[6,b]-->7
```

Result:

Implementation of a program for converting Regular Expression to NFA.

has been done successfully.