Compiler Design

Regex to NFA

EXPERIMENT - 2

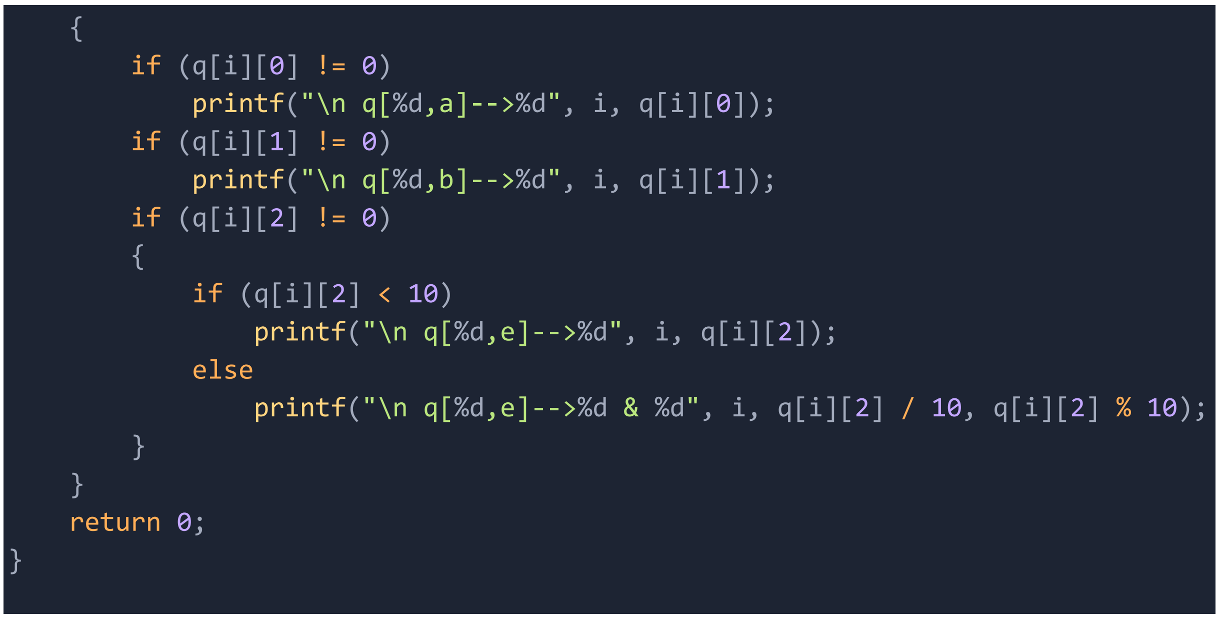
Roehit Ranganathan | RA1911033010017
1-28-2022

Aim:

To Build a program to covert regular expression into NFA in C/C++/Java.

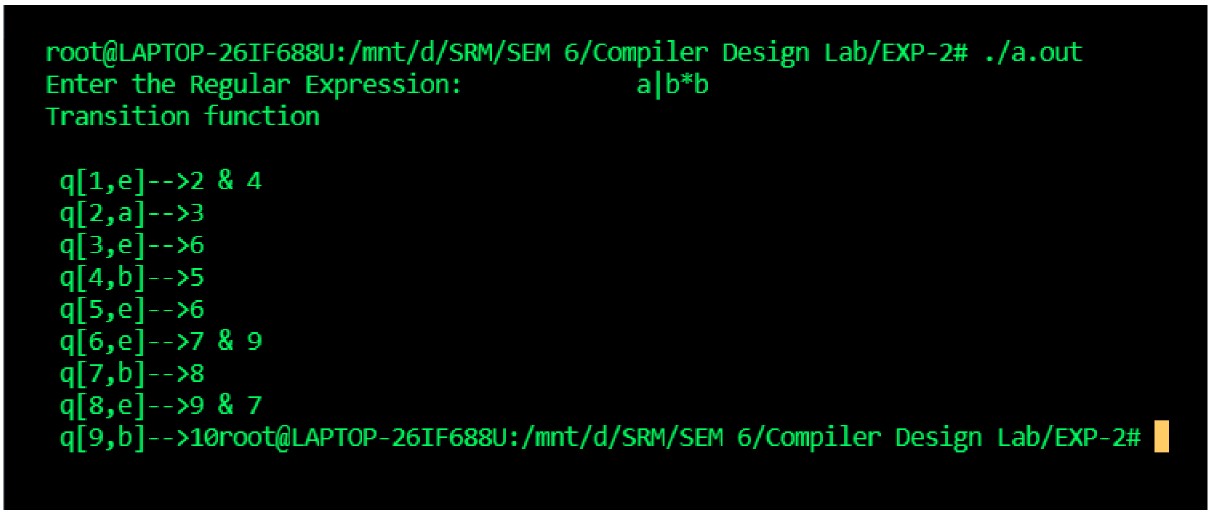
Program:

```c
#include <stdio.h>
#include <string.h>
int main()
{
    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("%s", "Enter the Regular Expression:            ");
    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++;
        }
        // 1
        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("Transition function \n");
    for (i = 0; i <= j; i++)
```



```c
    {
        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);
        }
    }
    return 0;
}
```

Output:



```
root@LAPTOP-26IF688U:/mnt/d/SRM/SEM 6/Compiler Design Lab/EXP-2# ./a.out
Enter the Regular Expression:            a|b*b
Transition function

 q[1,e]-->2 & 4
 q[2,a]-->3
 q[3,e]-->6
 q[4,b]-->5
 q[5,e]-->6
 q[6,e]-->7 & 9
 q[7,b]-->8
 q[8,e]-->9 & 7
 q[9,b]-->10root@LAPTOP-26IF688U:/mnt/d/SRM/SEM 6/Compiler Design Lab/EXP-2#
```

Result:

The Program was successfully compiled and run.