Experiment Number 2

Aim: Conversion of Regular Expression to NFA

Algorithm:

Step 1: Start

Step 2: Create start state.

Step 3: Add edge between previous and new states.

Step 4: For each operator create new state and add edges.

Step 5: Mark final state as accepting state when RE fully matched.

Step 6: Stop.

Code:

```
#include<stdio.h>
                    #include<string.h>
                    int main()
                    {
                           char reg[20]; int q[20][3],i=0,j=1,len,a,b;
                           for(a=0;a<20;a++) for(b=0;b<3;b++) q[a][b]=0;
                           scanf("%s",reg);
                           printf("Given regular expression: %s\n",reg);
                           len=strlen(reg);
                           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("\n\tTransition Table \n");
        printf("_____________________________________\n");
        printf("Current State |\tInput |\tNext State");
        printf("\n_____________________________________\n");
        for(i=0;i<=j;i++)
        {
                if(q[i][0]!=0) printf("\n  q[%d]\t      |   a   |
q[%d]",i,q[i][0]);
                if(q[i][1]!=0) printf("\n  q[%d]\t      |   b   |
q[%d]",i,q[i][1]);
                if(q[i][2]!=0)
                {
                        if(q[i][2]<10) printf("\n  q[%d]\t      |   e
|  q[%d]",i,q[i][2]);
                        else printf("\n  q[%d]\t      |   e   |  q[%d]
, q[%d]",i,q[i][2]/10,q[i][2]%10);
                }
        }
        printf("\n_____________________________________\n");
        return 0;
```

```
}
```

Output:

```
ab*
Given regular expression: ab*

        Transition Table

_____________________________________
Current State | Input | Next State
_____________________________________

  q[1]        |   a   |  q[2]
  q[2]        |   e   |  q[3] , q[5]
  q[3]        |   b   |  q[4]
  q[4]        |   e   |  q[5] , q[3]
_____________________________________
```

Result: Thus, RE to NFA implemented successfully.