REGULAR EXPRESSION TO NFA

NAME- ATHRESH KUMAR LABDE

REG NO- RA1911033010146

M1 (CSE-SE)

AIM: To convert given regular expression to NFA

ALGORITHM:

The algorithm works recursively by splitting an expression into its constituent subexpressions, from which the NFA will be constructed using a set of rules. More precisely, from a regular expression E, the obtained automaton A with the transition function which respects the following properties:

* A has exactly one initial state q_0, which is not accessible from any other state. That is, for any state q and any letter a, {\displaystyle \Delta (q,a)} does not contain q_0.

* A has exactly one final state q_f, which is not co-accessible from any other state. That is, for any letter a, {\displaystyle \Delta (q_{f},a)=\emptyset }.

* Let c be the number of concatenation of the regular expression E and let s be the number of symbols apart from parentheses — that is, $|$, *, a and ε. Then, the number of states of A is $2s - c$ (linear in the size of E).

* The number of transitions leaving any state is at most two.

* Since an NFA of m states and at most e transitions from each state can match a string of length n in time $O(emn)$, a Thompson NFA can do pattern matching in linear time, assuming a fixed-size alphabet.

CODE:

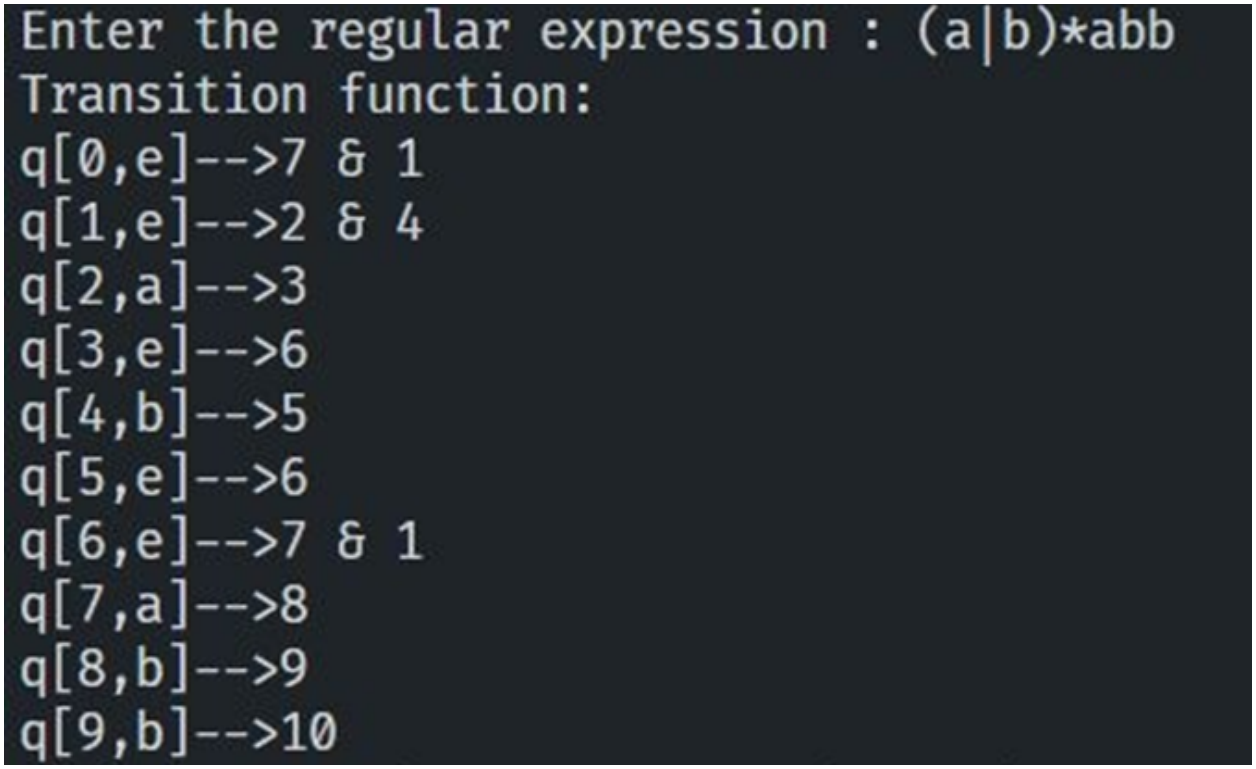
```
transition_table = [ [0]*3 for _ in range(20) ]

re = input("Enter the regular expression : ") re += " "

i = 0

j = 1 while(i<len(re)):
```

```
if re[i] == 'a': try:
if re[i+1] != '|' and re[i+1] !='*':
transition_table[j][0] = j+1 j += 1
elif re[i+1] == '|' and re[i+2] =='b': transition_table[j][2]=((j+1)*10)+(j+3) j+=1
transition_table[j][0]=j+1 j+=1
transition_table[j][2]=j+3 j+=1
transition_table[j][1]=j+1 j+=1
transition_table[j][2]=j+1 j+=1
i=i+2
elif re[i+1]=='*': transition_table[j][2]=((j+1)*10)+(j+3) j+=1
transition_table[j][0]=j+1 j+=1
transition_table[j][2]=((j+1)*10)+(j-1) j+=1
except:
transition_table[j][0] = j+1
elif re[i] == 'b': try:
if re[i+1] != '|' and re[i+1] !='*': transition_table[j][1] = j+1
j += 1
elif re[i+1]=='|' and re[i+2]=='a': transition_table[j][2]=((j+1)*10)+(j+3)

j+=1
transition_table[j][1]=j+1 j+=1
transition_table[j][2]=j+3 j+=1
transition_table[j][0]=j+1 j+=1
```

```
transition_table[j][2]=j+1 j+=1

i=i+2

elif re[i+1]=='*': transition_table[j][2]=((j+1)*10)+(j+3) j+=1

transition_table[j][1]=j+1 j+=1

transition_table[j][2]=((j+1)*10)+(j-1) j+=1

except:

transition_table[j][1] = j+1


elif re[i]=='e' and re[i+1]!='|'and re[i+1]!='*': transition_table[j][2]=j+1

j+=1

elif re[i]==')' and re[i+1]=='*':

transition_table[0][2]=((j+1)*10)+1 transition_table[j][2]=((j+1)*10)+1 j+=1

i +=1


print ("Transition function:") for i in range(j):

if(transition_table[i][0]!=0):

print("q[{0},a]-->{1}".format(i,transition_table[i][0])) if(transition_table[i][1]!=0):

print("q[{0},b]-->{1}".format(i,transition_table[i][1])) if(transition_table[i][2]!=0):

if(transition_table[i][2]<10):

print("q[{0},e]-->{1}".format(i,transition_table[i][2])) else:

print("q[{0},e]-->{1} &

{2}".format(i,int(transition_table[i][2]/10),transition_table[i][2]%10))
```

INPUT : (a|b)*abb

OUTPUT :



RESULT: The given program for conversion of a regular expression to NFA has been successfully executed.