4th

Experiment 3 : Regular Expression to NFA

Aim: The aim of this experiment is to study and implement a method to convert Regular Expression to NFA

Algorithm:

1. Start
2. Get the input from the user
3. Initialize seperate variables for NFA
4. According to Thompson construction method, convert RE to NFA
5. Display transition table for new NFA.
6. Stop.

Manual Working

1. Input : $(a+b)^*abb$

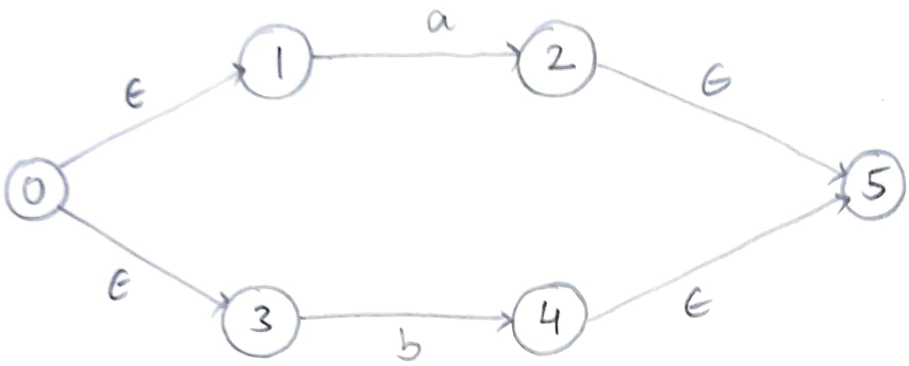
Output :

		a	b	∈
	0	∅	∅	{7, 1}
→	1	∅	∅	{2, 4}
	2	3	∅	∅
	3	∅	∅	6
	4	∅	5	∅
	5	∅	∅	6
	6	∅	∅	{7, 1}
	7	8	∅	∅
	8	∅	9	∅
	9	∅	10	∅
*	10	∅	∅	∅

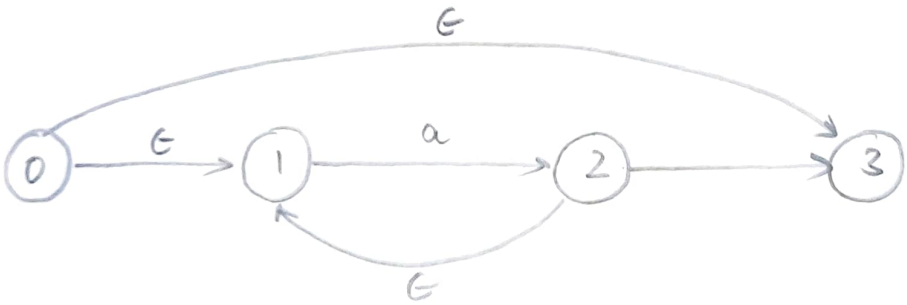
MANUAL WORKING

Thompson construction.

a b

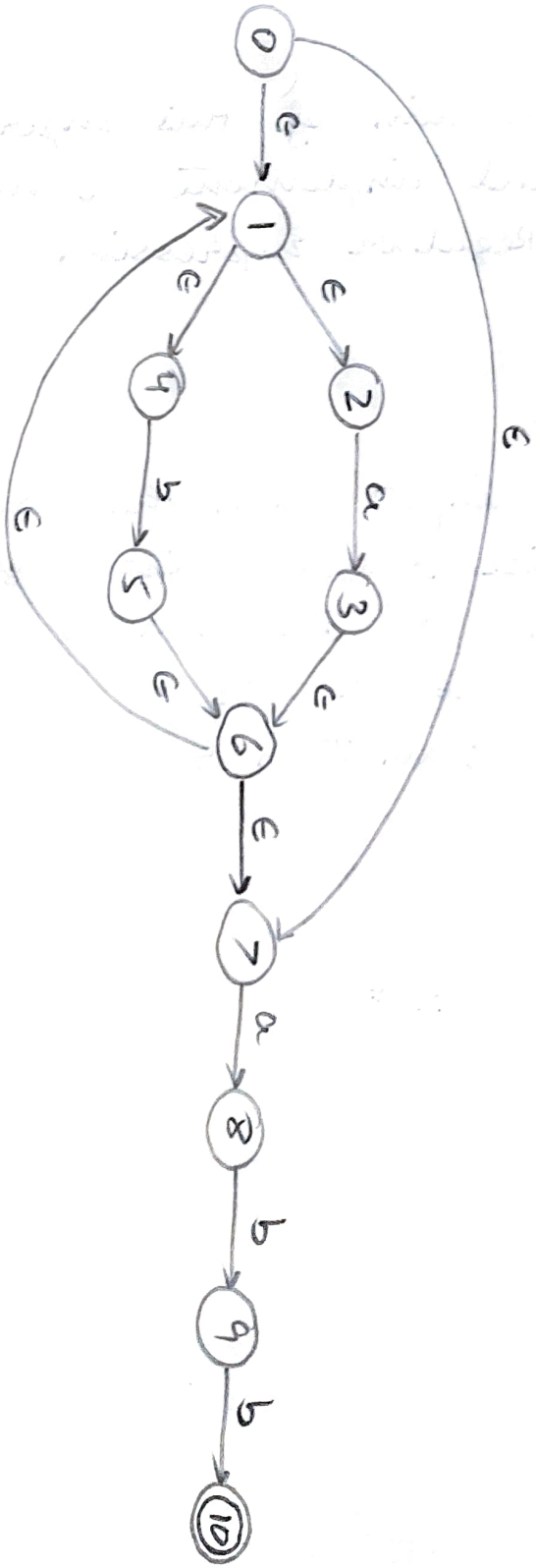


a + b



a^*

Input : $(a+b)^* abb$



Sashrika Surya
RA1911027010092
Section: N2
Lab Batch: 1

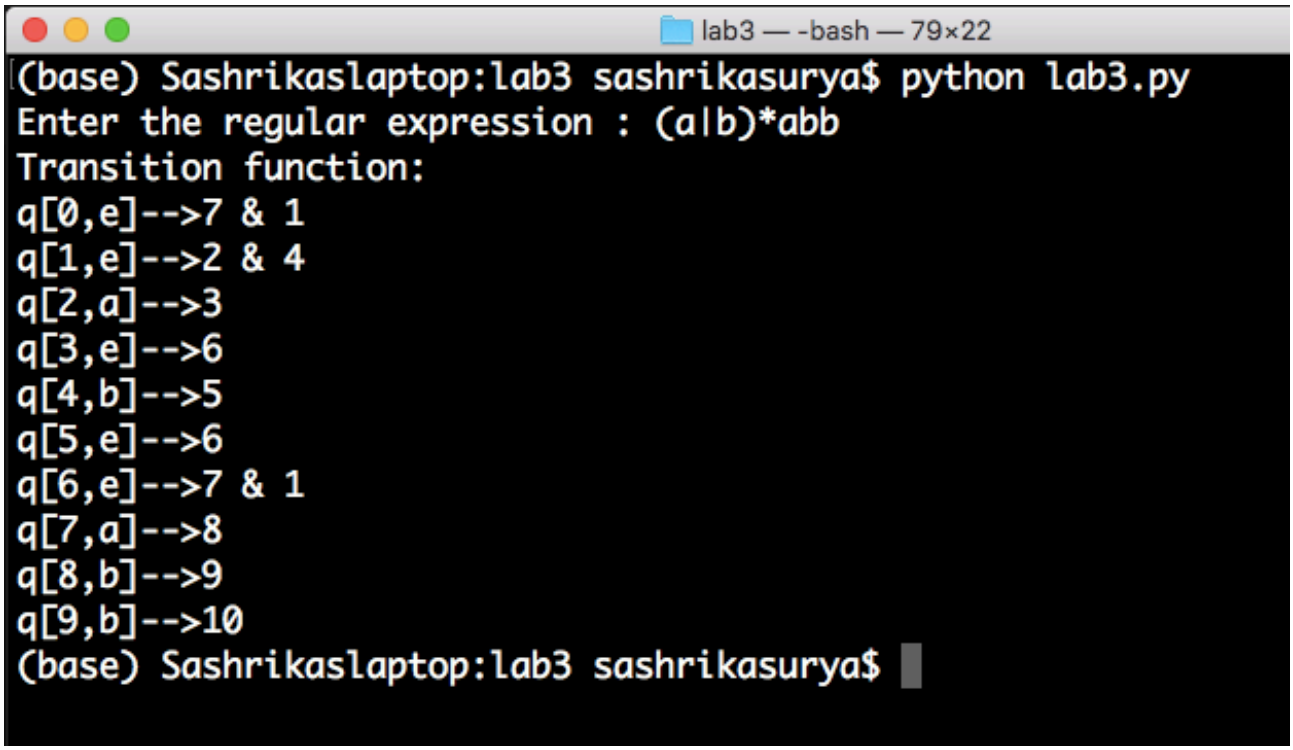
Compiler Design Lab
Experiment 3: RE to NFA

Code:

```
# RE TO NFA
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))
```

Output:


lab3 — -bash — 79×22
(base) Sashrikaslaptop:lab3 sashrikasurya$ python lab3.py
Enter the regular expression : (a|b)*abb
Transition function:
q[0,e]-->7 & 1
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 & 1
q[7,a]-->8
q[8,b]-->9
q[9,b]-->10
(base) Sashrikaslaptop:lab3 sashrikasurya$