**ABHIJIT DASH**

**RA1811031010112**

**REGULAR EXPRESSION TO NFA**

**AIM:-**

Conversion from Regular Expression to NFA.

**CODING:-**

#include<bits/stdc++.h>

using namespace std;

char reg[20];

void postfix();

void e\_nfa();

void disp(int,char,int);

int main()

{

cout<<"Enter Regular Expression:";

cin>>reg;

postfix();

e\_nfa();

return 0;

}

void postfix()

{

char string[10],stack[10];

int string\_n=0,stack\_n=0;

int n=0;

strcat(reg,"X");

while(reg[n]!='\0')

{

switch(reg[n])

{

case 'a': string [string\_n]='a';

string\_n++;

string[string\_n]='\0';

break;

case 'b': string[string\_n]='b';

string\_n++;

string[string\_n]='\0';

break;

case ' ' : string[string\_n]=' ';

string\_n++;

string[string\_n]='\0';

break;

case '(' : stack[stack\_n]='(';

stack\_n++;

break;

case ')' : stack\_n--;

while(stack[stack\_n]!='(')

{

string[string\_n]=stack[stack\_n];

stack[stack\_n]='\0';

string\_n++;

string[string\_n]='\0';

stack\_n--;

}

stack[stack\_n]='\0';

break;

case 'X' : while(stack\_n!=0)

{

stack\_n--;

string[string\_n]=stack[stack\_n];

stack[stack\_n]='\0';

string\_n++;

string[string\_n]='\0';

}

break;

case '+':

if(stack[stack\_n-1]!='+' && stack[stack\_n-1]!='.')

{

stack[stack\_n]='+';

stack\_n++;

stack[stack\_n]='\0';

break;

}

else

{

string[string\_n]=stack[stack\_n-1];

string\_n++;

stack[stack\_n-1]='+';

break;

}

case '.' :

if(stack[stack\_n-1]!='+'&&stack[stack\_n-1]!='.')

{

stack[stack\_n]='.';

stack\_n++;

stack[stack\_n]='\0';

break;

}

else

{

string[string\_n]=stack[stack\_n-1];

string\_n++;

stack[stack\_n-1]='.';

break;

}

default:break;

}

n++;

}

strcpy(reg,string);

}

void e\_nfa()

{

int strt[3],last[3],s,l;

int n=0,x=0,i=-1;

cout<<"\nTransitions:\n";

while(reg[n]!='\0')

{

switch(reg[n])

{

case 'a':i++;

strt[i]=x++;

last[i]=x++;

disp(strt[i],'a',last[i]);

break;

case 'b':i++;

strt[i]=x++;

last[i]=x++;

disp(strt[i],'b',last[i]);

break;

case '+' : s=x++;

l=x++;

disp(s,'e',strt[i]);

disp(s,'e',strt[i-1]);

disp(last[i],'e',l);

disp(last[i-1],'e',l);

i--;

strt[i]=s;

last[i]=l;

break;

case '.' : disp(last[i-1],'e',strt[i]);

last[i-1]=last[i];

i--;

break;

case '\*' : s=x++;

l=x++;

disp(s,'e',strt[i]);

disp(s,'e',l);

disp(last[i],'e',strt[i]);

disp(last[i],'e',l);

strt[i]=s;

last[i]=l;

break;

default:break;

}

n++;

}

cout<<i<<" "<<strt[i]<<" "<<last[i];

}

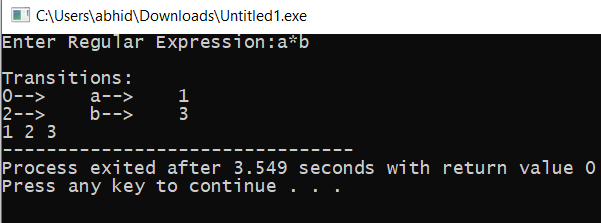
void disp(int qs,char a,int qf)

{

cout<<qs<<"-->\t"<<a<<"-->\t"<<qf<<"\n";

}

**OUTPUT SCREENSHOTS:-**



**RESULT:-**

Conversion from Regular Expression to NFA is successful.