CD EXP 2

Regular expression to NFA

AIM: To perform Regular expression to NFA conversion in any compiler.

ALGORITHM:

- “a” has exactly one final state qf, which is not co-accessible from any other state. That is, for any letter “a”.
- 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, a Thompson NFA can do pattern matching in linear time, assuming a fixed-size alphabet.

Code:

```
#include<iostream>
#include<stack>
#include<string>
#include <algorithm>
#include<vector>
using namespace std;
class node{
public:
    char input;
    int to;
    node *next;
};
int prec(char c){
    if(c=='*'){
        return 3;
    }else if(c=='.'){
        return 2;
    }else if(c=='+'){
        return 1;
    }else{
        return -1;
    }
}
```

```
string post(string s)
{
    stack<char> st;
    st.push('N');
    int l = s.length();
    string ns;
    for(int i = 0; i < l; i++)
    {
        if((s[i] >= 'a' && s[i] <= 'z')||(s[i] >= 'A' && s[i] <= 'Z')){
            ns+=s[i];
        }
        else if(s[i] == '('){
            st.push('(');
        }
        else if(s[i] == ')')
        {
            while(st.top() != 'N' && st.top() != '(')
            {
                char c = st.top();
                st.pop();
               ns += c;
            }
            if(st.top() == '(')
            {
                char c = st.top();
                st.pop();
            }
        }
        else{
            while(st.top() != 'N' && prec(s[i]) <= prec(st.top()))
            {
                char c = st.top();
                st.pop();
                ns += c;
            }
            st.push(s[i]);
        }
    }
    while(st.top() != 'N')
    {
        char c = st.top();
        st.pop();
        ns += c;
    }
return ns;
}
void printnode(vector<node*> v){
    cout<<"_____"<<endl;
    cout<<"| from state\t| input\t| tostates"<<endl;
    for(int i=0;i<v.size();i++){
        cout<<"| "<<i<<"          \t|";
        node* head = v[i];
        cout<<head->input;
        bool first = true;
        while(head!=NULL){
            if (first)
            {
                cout<<"     \t|";
                first = false;
            }else{
                cout<<"     \t";
```

```
            }
            cout<<head->to;
            head = head->next;
        }
        cout<<endl;

    }
    cout<<"_____"<<endl;
}
node* makenode(char in){
    node* a = new node;
    a->input = in;
    a->to = -1;
    a->next = NULL;
    return a;
}
node* copynode(node* a){
    node* b = new node;
    b->input = -1;
    b->to = -1;
    b->next =NULL;
    return b;
}
void andd(vector<node*> &v,vector<vector<int> > &st){
    int x,y;
    int first,last1;
    y = st[st.size()-1][0];
    x = st[st.size()-2][1];
    first = st[st.size()-2][0];
    last1 = st[st.size()-1][1];
    st.pop_back();
    st.pop_back();
    vector<int> ptemp;
    ptemp.push_back(first);
    ptemp.push_back(last1);
    st.push_back(ptemp);
    node* last = v[y];
    node * lnode= v[x];
    node* temp = copynode(last);
    // temp->to = -1;
    while(lnode->next!=NULL){
        lnode = lnode->next;
    }
    lnode->next = temp;
    lnode->to = y;
}
void orr(vector<node*> &v,vector<vector<int> > &st){
    int x,y,x1,y1;
    x = st[st.size()-2][0];
    y = st[st.size()-1][0];
    x1 = st[st.size()-2][1];
    y1 = st[st.size()-1][1];
    node* start = makenode('e');
    node* end = makenode('e');
    v.push_back(start);
    int firstnode = v.size() -1;
    v.push_back(end);
    int endnode = v.size() -1;
    st.pop_back();
    st.pop_back();
    vector<int> ptemp;
```

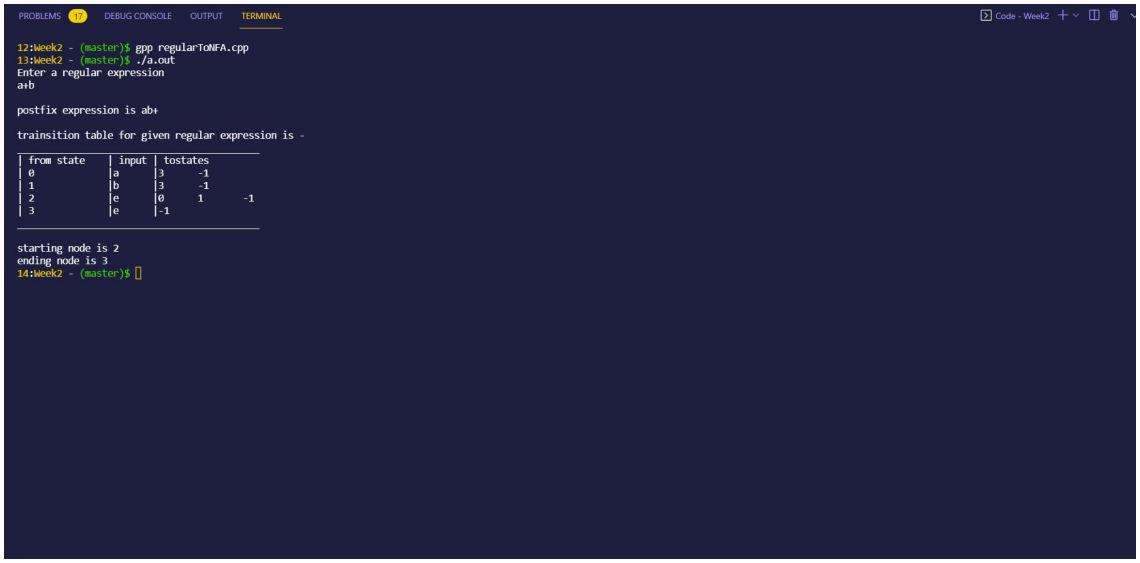
```
    ptemp.push_back(firstnode);
    ptemp.push_back(endnode);
    st.push_back(ptemp);
    for(int i=0;i<v.size()-2;i++){
        node* h=v[i];
        while(h->next!=NULL){
            if(h->to==x || h->to == y){
                h->to = firstnode;
            }
            h = h->next;
        }
    }
    node* temp = copynode(v[x]);
    node* temp1 = copynode(v[y]);
    node* t = v[firstnode];
    while(t->next!=NULL){
        t = t->next;
    }
    t->to = x;
    t->next  = temp;
    t->next->to = y;
    t->next->next = temp1;
    node* adlink = v[x1];
    while(adlink->next!=NULL){
        adlink = adlink->next;
    }
    adlink->to= endnode;
    adlink->next = copynode(end);
    node* adlink1 = v[y1];
    while(adlink1->next!=NULL){
        adlink1 = adlink1->next;
    }
    adlink1->to = endnode;
    adlink1->next = copynode(end);
}
void closure(vector<node*> &v, vector<vector<int> > &st){
    int x,x1;
    x = st[st.size()-1][0];
    x1 = st[st.size()-1][1];
    node* s = makenode('e');
    v.push_back(s);
    int firstnode = v.size() -1;
    st.pop_back();
    vector<int> ptemp;
    ptemp.push_back(x);
    ptemp.push_back(firstnode);
    st.push_back(ptemp);
    for(int i=0;i<v.size()-2;i++){
        node* h=v[i];
        while(h->next!=NULL){
            if(h->to==x){
                h->to = firstnode;
            }
            h = h->next;
        }
    }




    node* t = v[x1];
```

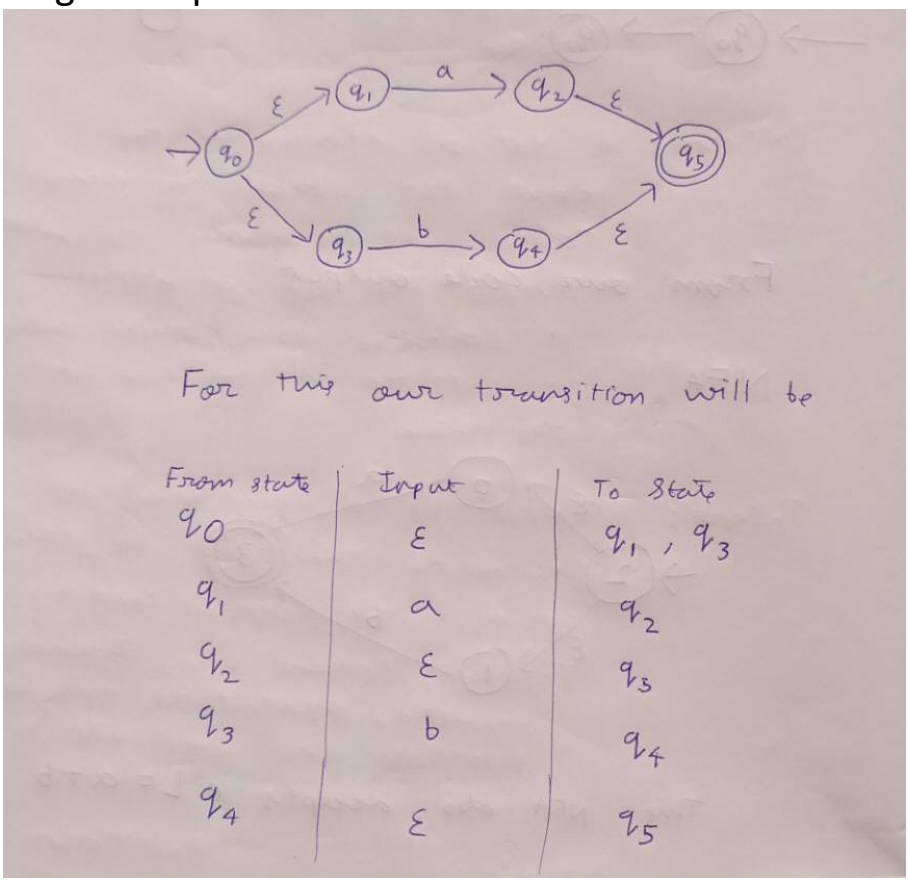
```
    while(t->next!=NULL){
        t = t->next;
    }
    t->to = x;
    t->next = copynode(t);
    t->next->to = firstnode;
    t->next->next = copynode(s);
}
int main(){
    string in;
    cout<<"Enter a regular expression\n";
    cin>>in;
    string o;
    vector<node*> v;
    o = post(in);
    cout<<"\npostfix expression is "<< o<<endl;
    vector<vector<int>> st;
    int firstnode = 0;
    for(int l =0 ;l<o.length();l++){
        if(o[l] !='+' && o[l]!='*' && o[l]!='.'){
            node* temp = makenode(o[l]);
            v.push_back(temp);
            vector<int> ptemp;
            ptemp.push_back(v.size()-1);
            ptemp.push_back(v.size()-1);
            st.push_back(ptemp);
        }
        else if(o[l]=='.'){
            andd(v,st);
        }
        else if(o[l]=='+'){
            orr(v,st);
        }
        else if(o[l]=='*'){
            closure(v,st);
        }
    }
    cout<<"\ntrainsition table for given regular expression is - \n";
    printnode(v);
    cout<<endl;
    cout<<"starting node is ";
    cout<<st[st.size()-1][0]<<endl;
    cout<<"ending node is ";
    cout<<st[st.size()-1][1]<<endl;
    return 0;
}
```

Output:

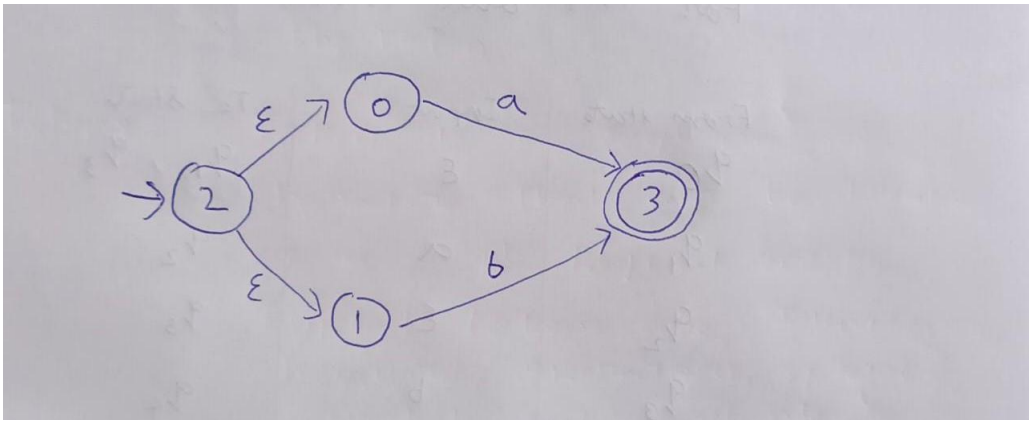

PROBLEMS 17
DEBUG CONSOLE
OUTPUT
TERMINAL
Code - Week2
12:Week2 - (master)$ gpp regularToNFA.cpp
13:Week2 - (master)$ ./a.out
Enter a regular expression
a+b
postfix expression is ab+
trainsition table for given regular expression is -
| from state | input | tostates
| 0 |a |3 -1
| 1 |b |3 -1
| 2 |e |0 1 -1
| 3 |e |-1
starting node is 2
ending node is 3
14:Week2 - (master)$


14:Week2 - (master)$ gpp regularToNFA.cpp
15:Week2 - (master)$ ./a.out
Enter a regular expression
ab
postfix expression is ab
trainsition table for given regular expression is -
| from state | input | tostates
| 0 |a |-1
| 1 |b |-1
starting node is 1
ending node is 1
16:Week2 - (master)$ ./a.out
Enter a regular expression
a|b
postfix expression is ab|
trainsition table for given regular expression is -
| from state | input | tostates
| 0 |a |-1
| 1 |b |-1
| 2 ||| |-1
starting node is 2
ending node is 2

1) Regular Expression for L = a+b



Here start state is q0.
Here final state is q5.



Result:

Hence Regular expression was converted to NFA.