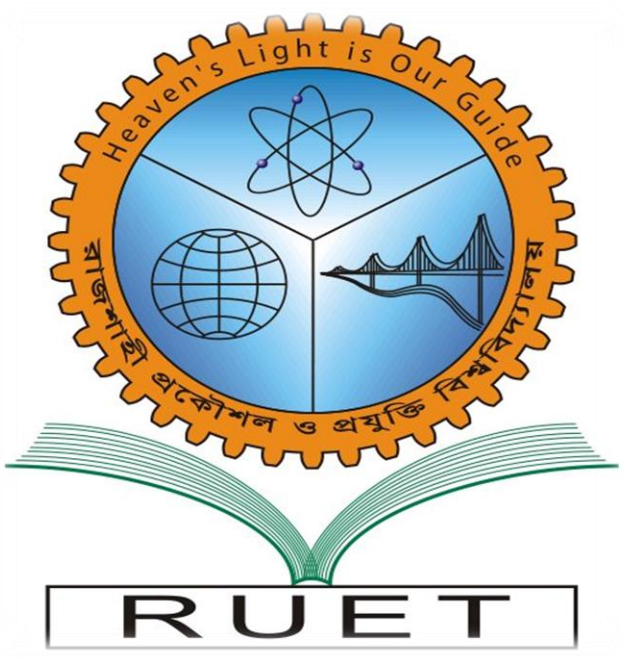


Lab 4

Course Title : Sessional Based on CSE 2205.

Course No. : CSE 2206.

Submitted by

Name: Md. Shabir Khan Akash.

Class: 2nd year, even semester.

Department : CSE

Roll: 1603108

Section: B

Submitted to

Suhrid Shakhar Ghosh

Lecturer

Department of CSE, RUET

Name of the Experiment:

Conversion of DFA to Regular Expression (RegEx).

Theory:

We can convert a DFA to a regular expression by an inductive construction in which expressions for the labels of paths allowed to pass through increasingly larger sets of states are constructed. Alternatively, we can use a state elimination procedure to build the regular expression for a DFA. In the other direction. We can construct recursively an e-NFA from regular expressions, and then convert the e-NFA to a DFA ,if we wish. The approach to constructing regular expressions that we shall now learn involves eliminating states. When we eliminate a states ,all the paths that went through s no longer existing the automaton. If the language of the automation is not to change, we must include, on an arc that goes directly from q to p, the labels of paths that went from state q to state p, through s. Since the label of this arc may now involve string s, rather than single symbols, and the re may even be an infinite number of such strings, we can not simply list the strings as a label. Fortunately, there is a simple, finite way to represent all such strings: use are regular expression. This conversion can be executed by using BFS or DFS. Breadth-first search (BFS) is an algorithm for traversing or searching tree or graph data structures. It starts at the tree root (or some arbitrary node of a graph, sometimes referred to as a 'search key'), and explores all of the neighbor nodes at the present depth prior to moving on to the nodes at the next depth level. Depth-first search (DFS) is an algorithm for traversing or searching tree or graph data structures. The algorithm starts at the root node (selecting some arbitrary node as the root node in the case of a graph) and explores as far as possible along each branch before backtracking.

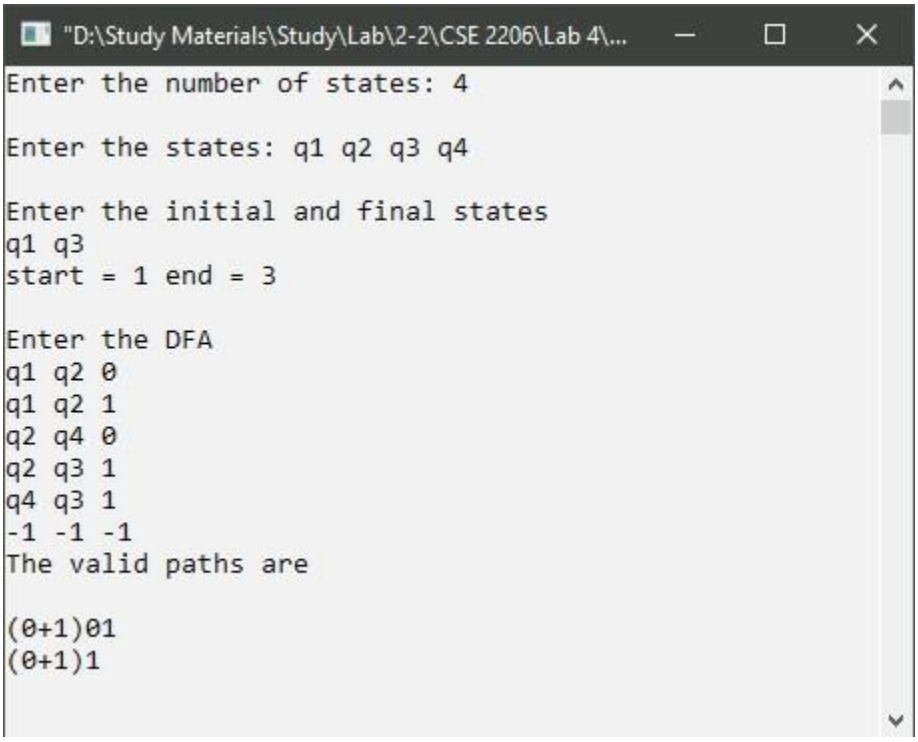
Code :

```
#include<bits/stdc++.h>
using namespace std;
#define ll long long

vector<string>edge[10];
vector<ll>graph[10];
map<string,ll>mm;
ll cnt=0;
string arr[10];
string path[10];

ll start,finish;
```

```
void dfs(ll s,string str)
{
   if(s==start)str.clear();
   if(s==finish)
   {
      path[++cnt]=str;
      return;
   }
   string aaa=str;
   for(ll i=0;i<graph[s].size();i++)
   {
      ll v=graph[s][i];
      str=aaa+edge[s][v];
      dfs(v,str);
   }
}



int main()
{
   for(ll i=0;i<9;i++)
   {
      edge[i].resize(10);
   }
   cout<<"Enter the number of states: ";
   ll n;
   cin>>n;
   cout<<"\nEnter the states: ";
   for(ll i=1;i<=n;i++)
   {
      cin>>arr[i];
      if(mm[arr[i]]==0)
      {
         mm[arr[i]]=++cnt;
      }
   }
   cout<<"\nEnter the initial and final states"<<endl;
   string s,e;
   cin>>s>>e;
   ll st=mm[s];
   ll en=mm[e];
   cout<<"start = "<<st<<" end = "<<en<<endl;
   cout<<"\nEnter the DFA"<<endl;
   while(1)
   {
      string str;
      cin>>s>>e>>str;
      if(s=="-1")break;
      ll a=mm[s],b=mm[e];
      if(edge[a][b].empty()){
```

```
            graph[a].push_back(b);
            edge[a][b]=str;
        }
        else
        {
            edge[a][b]+="+";
            edge[a][b]+=str;
            str=edge[a][b];
            string s;
            s.push_back('(');
            for(ll i=0;i<str.size();i++)
                s.push_back(str[i]);
            s.push_back(')');
            edge[a][b]=s;

        }
    }
    start=st;
    finish=en;
    cnt=0;
    string aa;

    dfs(start,aa);

    cout<<"The valid paths are"<<endl<<endl;
    for(ll i=1;i<=cnt;i++)
    {
        cout<<path[i]<<endl;
    }

    return 0;
}
```

Output:



Discussion:

In the above code, conversion of DFA into Regular Expression was performed by using DFS. DFS is a graph traversing algorithm which stands for Depth First Search. In the above code, for inputs, there was an integer that indicates the number of states are present in the DFA. Then after entering the states, the initial and the final states were given as inputs. Then the transition table of the following DFA was given as input. Then graph[int a] and Edge[int a][int b] was declared and handled. Then after applying DFS algorithm in the following graph , the expected path was found which is the expected Regular Expression.