EX NO 3

Conversion of NFA to DFA

AIM: To write a program for converting NFA to DFA

CODE:

```
#include <bits/stdc++.h>
#define STATES 50
using namespace std;
struct Dstate
{
char name;
char StateString[STATES+1];
char trans[10];
int is_final;
}Dstates[50];
struct tran
{
char sym;
int tostates[50];
int notran;
};
struct state
{
int no;
struct tran tranlist[50];
};
int stackA[100],stackB[100],C[100],Cptr=-1,Aptr=-1,Bptr=-1;
struct state States[STATES];
char temp[STATES+1],inp[10];
int nos,noi,nof,j,k,nods=-1;
void pushA(int z)
{
stackA[++Aptr]=z;
}
void pushB(int z)
{
stackB[++Bptr]=z;
}
int popA()
{
return stackA[Aptr--];
}
```

```
void copy(int i)
{
char temp[STATES+1]=" ";
int k=0;
Bptr=-1;
strcpy(temp,Dstates[i].StateString);
while(temp[k]!='\0')
{
pushB(temp[k]-'0');
k++;
}
}
int popB()
{
return stackB[Bptr--];
}
int peekB()
{
return stackA[Bptr];
}
int peekA()
{
return stackA[Aptr];
}
int seek(int arr[],int ptr,int s)
{
int i;
for(i=0;i<=ptr;i++)
if(s==arr[i]) return 1;
return 0;
}
void sort()
{
int i,j,temp;
for(i=0;i<Bptr;i++)
for(j=0;j<(Bptr-i);j++)
if(stackB[j]>stackB[j+1])
{
temp=stackB[j];
stackB[j]=stackB[j+1];
stackB[j+1]=temp;
}
}
void tostring()
```

```
{
int i=0;
sort();
for(i=0;i<=Bptr;i++)
temp[i]=stackB[i]+'0';
temp[i]='\0';
}
void display_DTran()
{
int i,j;
cout<<"\n\t\t DFA Transition Table ";
cout<<"\n\t\t ---------- ";
cout<<"\nStates\tString\tInputs\n ";
for(i=0;i<noi;i++)
cout<<"\t"<<inp[i];
cout<<"\n \t----------";
for(i=0;i<nods;i++)
{
if(Dstates[i].is_final==0)cout<<"\n"<<Dstates[i].name;
else cout<<"\n"<<Dstates[i].name;
cout<<"\t"<<Dstates[i].StateString;
for(j=0;j<noi;j++)
cout<<"\t"<<Dstates[i].trans[j];
}
cout<<"\n";
}
void move(int st,int j)
{
int ctr=0;
while(ctr<States[st].tranlist[j].notran)
{
pushA(States[st].tranlist[j].tostates[ctr++]);
}
}
void lambda_closure(int st)
{
int ctr=0,in_state=st,curst=st,chk;
while(Aptr!=-1)
{
curst=popA();
ctr=0;
in_state=curst;
while(ctr<=States[curst].tranlist[noi].notran)
{
```

```
 chk=seek(stackB,Bptr,in_state);
 if(chk==0)
pushB(in_state);
 in_state=States[curst].tranlist[noi].tostates[ctr++];
 chk=seek(stackA,Aptr,in_state);
 if(chk==0 && ctr<=States[curst].tranlist[noi].notran)
pushA(in_state);
}
}
}
int main()
{
int final[20],start,fin=0,i;
char c,ans,st[20];
cout<<"\nEnter no. of states in NFA : ";
cin>>nos;
for(i=0;i<nos;i++)
States[i].no=i;
cout<<"\nEnter the start state : ";
cin>>start;
cout<<"Enter the no. of final states : ";
cin>>nof;
cout<<"\nEnter the final states : \n";
for(i=0;i<nof;i++)
cin>>final[i];
cout<<"\nEnter the no. of input symbols : ";
cin>>noi;
c=getchar();
cout<<"\nEnter the input symbols : \n ";
for(i=0;i<noi;i++)
{
cin>>inp[i];
c=getchar();
}
inp[i]='e';
cout<<"\nEnter the transitions : (-1 to stop)\n";
for(i=0;i<nos;i++)
{
for(j=0;j<=noi;j++)
{
States[i].tranlist[j].sym=inp[j];
k=0;
ans='y';
while(ans=='y')
```

```
{
cout<<"move("<<i<<","<<inp[j]<< ") :";
cin>>States[i].tranlist[j].tostates[k++];
if(States[i].tranlist[j].tostates[k-1]==-1)
{
k--; ans='n';
break;
}
}
States[i].tranlist[j].notran=k;
}
}
i=0;nods=0;fin=0;
pushA(start);
lambda_closure(peekA());
tostring();
Dstates[nods].name='A';
nods++;
strcpy(Dstates[0].StateString,temp);
while(i<nods)
{
for(j=0;j<noi;j++)
{
fin=0;
copy(i);
while(Bptr!=-1)
{
move(popB(),j);
}
while(Aptr!=-1)
lambda_closure(peekA());
tostring();
for(k=0;k<nods;k++)
{
if((strcmp(temp,Dstates[k].StateString)==0))
{
Dstates[i].trans[j]=Dstates[k].name;
break;
}
}
if(k==nods)
{
nods++;
for(k=0;k<nof;k++)
```

```
{
fin=seek(stackB,Bptr,final[k]);
if(fin==1)
{
Dstates[nods-1].is_final=1;
break;
}
}
strcpy(Dstates[nods-1].StateString,temp);
Dstates[nods-1].name='A'+nods-1;
Dstates[i].trans[j]=Dstates[nods-1].name;
}
}
i++;
}
display_DTran();
}
```

INPUT:

```
Enter no. of states in NFA : 3

Enter the start state : 0
Enter the no. of final states : 1

Enter the final states :
2

Enter the no. of input symbols : 2

Enter the input symbols :
 A B

Enter the transitions : (-1 to stop)
move(0,A) :0
move(0,A) :1
move(0,A) :-1
move(0,B) :-1
move(0,e) :-1
move(1,A) :-1
move(1,B) :2
move(1,B) :-1
move(1,e) :-1
move(2,A) :-1
move(2,B) :-1
move(2,e) :-1
```

OUTPUT:

```
                DFA Transition Table
                --------------------
States  String  Inputs
        A       B
        ----------
A       0       B       C
B       01      B       D
C               C       C
D       2       C       C
```

RESULT: The given NFA was successfully converted to a DFA.