Compiler Design Lab

NAME: Priyanka Srinivas
REGISTER NUMBER: RA1911026010014
SECTION: K1

Exercise 3: Conversion of NFA to DFA

AIM: To write a program to implement the conversion of NFA to DFA

ALGORITHM:

1. Start the program.
2. Construct the transition table of the given NFA machine.
3. Scan the next states column in the transition table from initial state to final state.
4. If any of the next states consists of more than one state on the single input alphabet. Then merge them and make it a new state. Place this newly constructed state in the DFA transition table as a present state.
5. The next state of this newly constructed state on the input alphabet will be the summation of each next state which parts in the NFA transition table.
6. Repeat step 3 to step 5 until all the states in the NFA transition table will be scanned completely.
7. The final transition table must have a single next state at a single input alphabet.

PROGRAM:

```
#include <stdio.h>
int main()
{
    int nfa[5][2];
    nfa[1][1]=12;
    nfa[1][2]=1;
    nfa[2][1]=0;
    nfa[2][2]=3;
    nfa[3][1]=0;
    nfa[3][2]=4;
    nfa[4][1]=0;
    nfa[4][2]=0;
    int dfa[10][2];
    int dstate[10];
    int i=1, n, j, k, flag=0, m, q, r;
    dstate[i++]=1;
```

```
n=i;

dfa[1][1]=nfa[1][1];
dfa[1][2]=nfa[1][2];
printf("\nf(%d,a)=%d",dstate[1],dfa[1][1]);
printf("\nf(%d,b)=%d",dstate[1],dfa[1][2]);

for(j=1;j<n;j++)
{
    if(dfa[1][1]!=dstate[j])
      flag++;
}
if(flag==n-1)
{
    dstate[i++]=dfa[1][1];
    n++;
}
flag=0;
for(j=1;j<n;j++)
{
     if(dfa[1][2]!=dstate[j])
        flag++;
}
if(flag==n-1)
{
     dstate[i++]=dfa[1][2];
     n++;
}
k=2;
while(dstate[k]!=0)
{
    m=dstate[k];
    if(m>10)
    {
        q=m/10;
        r=m%10;
    }
    if(nfa[r][1]!=0)
        dfa[k][1]=nfa[q][1]*10+nfa[r][1];
    else
        dfa[k][1]=nfa[q][1];
    if(nfa[r][2]!=0)
        dfa[k][2]=nfa[q][2]*10+nfa[r][2];
    else
        dfa[k][2]=nfa[q][2];
```

```
        printf("\nf(%d,a)=%d",dstate[k],dfa[k][1]);
        printf("\nf(%d,b)=%d",dstate[k],dfa[k][2]);

        flag=0;
        for(j=1;j<n;j++)
        {
            if(dfa[k][1]!=dstate[j])
                flag++;
        }
        if(flag==n-1)
        {
            dstate[i++]=dfa[k][1];
            n++;
        }
        flag=0;
        for(j=1;j<n;j++)
        {
            if(dfa[k][2]!=dstate[j])
                flag++;
        }
        if(flag==n-1)
        {
            dstate[i++]=dfa[k][2];
            n++;
        }
        k++;
    }
    return 0;
}
```

INPUT:

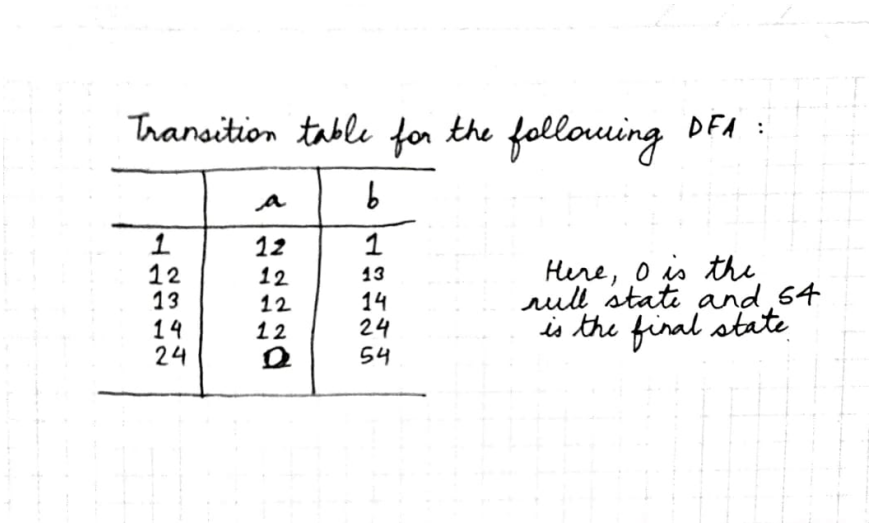
For NFA the input has been taken -

- NFA[1][1] signifies NFA(State 1, input a)
- NFA[1][2] signifies NFA(State 1, input b)
- NFA[2][1] signifies NFA(State 2, input a)
- NFA[2][2] signifies NFA(State 2, input b)
- NFA[3][1] signifies NFA(State 3, input a)
- NFA[3][2] signifies NFA(State 3, input b)
- NFA[4][1] signifies NFA(State 4, input a)
- NFA[4][2] signifies NFA(State 4, input b)

OUTPUT:

```
Command Prompt

C:\Users\priya\Desktop\Coding\compiler-design-lab>cd lab-3

C:\Users\priya\Desktop\Coding\compiler-design-lab\lab-3>gcc program.c -o output.exe

C:\Users\priya\Desktop\Coding\compiler-design-lab\lab-3>./output
'.' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\priya\Desktop\Coding\compiler-design-lab\lab-3>/output
'/output' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\priya\Desktop\Coding\compiler-design-lab\lab-3>output

f(1,a)=12
f(1,b)=1
f(12,a)=12
f(12,b)=13
f(13,a)=12
f(13,b)=14
f(14,a)=12
f(14,b)=24
f(24,a)=0
f(24,b)=54
C:\Users\priya\Desktop\Coding\compiler-design-lab\lab-3>
```

TRANSITION TABLE:



RESULT:

An NFA was converted to DFA and verified successfully using a transition table.