Experiment Number 3

Aim: Conversion from NFA to DFA.

Algorithm:

```
Step 1: Start
Step 2: Get number of final states.
Step 3: Get initial and transition states.
Step 4: Merge similar states and drop null states.
Step 5: Print respective output.
Step 6: Stop.
```

Code:

```
#include<stdio.h>
                        #include<string.h>
                        #include<math.h>

                        int ninputs;
                        int dfa[100][2][100] = {0};
                        int state[10000] = {0};
                        char ch[10], str[1000];
                        int go[10000][2] = {0};
                        int arr[10000] = {0};

                        int main()
                        {
                            int st, fin, in;
                            int f[10];
                            int i,j=3,s=0,final=0,flag=0,curr1,curr2,k,l;
                            int c;

                            printf("\nFollow the one based indexing\n");
```

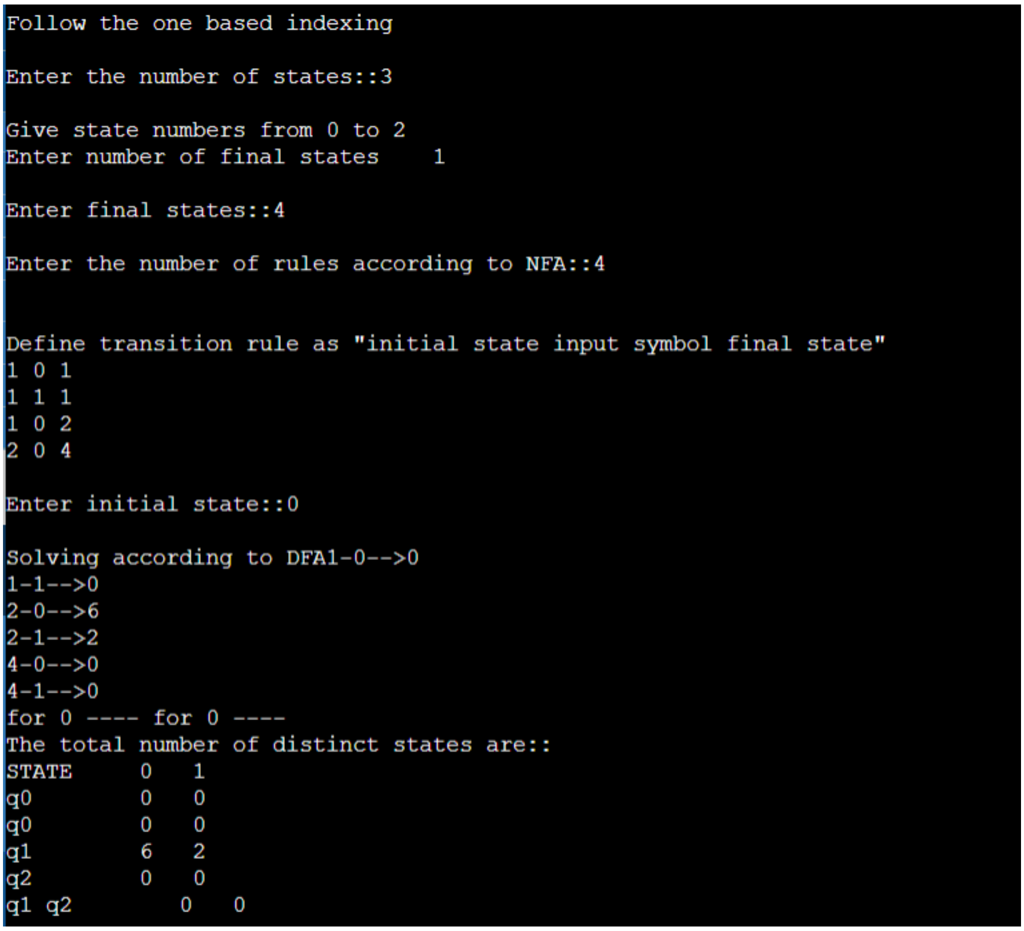
```
    printf("\nEnter the number of states::");
    scanf("%d",&st);

    printf("\nGive state numbers from 0 to %d",st-1);

    for(i=0;i<st;i++)
                state[(int)(pow(2,i))] = 1;

    printf("\nEnter number of final states\t");
    scanf("%d",&fin);

    printf("\nEnter final states::");
    for(i=0;i<fin;i++)
    {
        scanf("%d",&f[i]);
    }

    int p,q,r,rel;

    printf("\nEnter the number of rules according to NFA::");
    scanf("%d",&rel);

    printf("\n\nDefine transition rule as \"initial state input symbol final
state\"\n");



    for(i=0; i<rel; i++)
    {
```

```
    scanf("%d%d%d",&p,&q,&r);
              if (q==0)
                    dfa[p][0][r] = 1;
              else
                    dfa[p][1][r] = 1;
  }

  printf("\nEnter initial state::");
  scanf("%d",&in);

  in = pow(2,in);

  i=0;

  printf("\nSolving according to DFA");

  int x=0;
  for(i=0;i<st;i++)
  {
            for(j=0;j<2;j++)
            {
                        int stf=0;
                        for(k=0;k<st;k++)
                        {
                                    if(dfa[i][j][k]==1)
                                          stf = stf +
pow(2,k);
                        }
```

```
                                go[(int)(pow(2,i))][j] = stf;
                                printf("%d-%d--
>%d\n",(int)(pow(2,i)),j,stf);
                                if(state[stf]==0)
                                        arr[x++] = stf;
                                state[stf] = 1;
                }

    }



    //for new states
    for(i=0;i<x;i++)
    {
                printf("for %d ---- ",arr[x]);
                for(j=0;j<2;j++)
                {
                                int new=0;
                                for(k=0;k<st;k++)
                                {
                                                if(arr[i] & (1<<k))
                                                {
                                                                int h =
pow(2,k);


        if(new==0)

        new = go[h][j];
                                                                new = new
| (go[h][j]);
```

```
                        }
                }

                if(state[new]==0)
                {
                    arr[x++] = new;
                    state[new] = 1;
                }
        }
}

printf("\nThe total number of distinct states are::\n");

printf("STATE     0   1\n");

for(i=0;i<10000;i++)
{
        if(state[i]==1)
        {
                //printf("%d**",i);
                int y=0;
                if(i==0)
                    printf("q0 ");

                else
                for(j=0;j<st;j++)
                {
                        int x = 1<<j;
                        if(x&i)
                        {
                            printf("q%d ",j);
                            y = y+pow(2,j);
                            //printf("y=%d
",y);
                        }
                }
                //printf("%d",y);
                printf("      %d   %d",go[y][0],go[y][1]);
                printf("\n");
        }
  }




  j=3;
  while(j--)
  {
        printf("\nEnter string");
            scanf("%s",str);
            l = strlen(str);
            curr1 = in;
            flag = 0;
            printf("\nString takes the following path-->\n");
            printf("%d-",curr1);

            for(i=0;i<l;i++)
            {
                curr1 = go[curr1][str[i]-'0'];
                printf("%d-",curr1);
            }

            printf("\nFinal state - %d\n",curr1);

            for(i=0;i<fin;i++)
            {
                    if(curr1 & (1<<f[i]))
                    {
                            flag = 1;
                            break;
                    }
            }

            if(flag)
                printf("\nString Accepted");
            else
                printf("\nString Rejected");

     }



    return 0;
}
```

Output:



```
Follow the one based indexing

Enter the number of states::3

Give state numbers from 0 to 2
Enter number of final states    1

Enter final states::4

Enter the number of rules according to NFA::4


Define transition rule as "initial state input symbol final state"
1 0 1
1 1 1
1 0 2
2 0 4

Enter initial state::0

Solving according to DFA1-0-->0
1-1-->0
2-0-->6
2-1-->2
4-0-->0
4-1-->0
for 0 ---- for 0 ----
The total number of distinct states are::
STATE     0   1
q0        0   0
q0        0   0
q1        6   2
q2        0   0
q1 q2        0   0
```

Result: Thus, NFA to DFA implemented successfully.