```c
#include<stdio.h>
#include<stdlib.h>
struct node
{
        int st;
        struct node *link;
};

void findclosure(int,int);
void insert_trantbl(int ,char, int);
int findalpha(char);
void findfinalstate(void);
void unionclosure(int);
void print_e_closure(int);
static int
set[20],nostate,noalpha,s,notransition,nofinal,start,finalstate[20],c,r,buffer[20];
char alphabet[20];
static int e_closure[20][20]={0};
struct node * transition[20][20]={NULL};
void main()
{
           int i,j,k,m,t,n;

           struct node *temp;
           printf("enter the number of alphabets?\n");
           scanf("%d",&noalpha);
           getchar();
           printf("NOTE:- [ use letter e as epsilon]\n");

          printf("NOTE:- [e must be last character ,if it is present]\n");

          printf("\nEnter alphabets?\n");
          for(i=0;i<noalpha;i++)
         {

                  alphabet[i]=getchar();
                  getchar();
        }
        printf("Enter the number of states?\n");
        scanf("%d",&nostate);
        printf("Enter the start state?\n");
        scanf("%d",&start);
        printf("Enter the number of final states?\n");
        scanf("%d",&nofinal);
        printf("Enter the final states?\n");
        for(i=0;i<nofinal;i++)
                scanf("%d",&finalstate[i]);
         printf("Enter no of transition?\n");
        scanf("%d",&notransition);
        printf("NOTE:- [Transition is in the form--> qno   alphabet   qno]
\n",notransition);
        printf("NOTE:- [States number must be greater than zero]\n");
        printf("\nEnter transition?\n");
        for(i=0;i<notransition;i++)
        {


                scanf("%d %c%d",&r,&c,&s);
                insert_trantbl(r,c,s);

        }

        printf("\n");

        for(i=1;i<=nostate;i++)
        {
```

```
                c=0;
                for(j=0;j<20;j++)

                {
                                buffer[j]=0;
                                 e_closure[i][j]=0;
                }
                findclosure(i,i);
        }
        printf("Equivalent NFA without epsilon\n");
        printf("-------------------------------\n");
        printf("start state:");
        print_e_closure(start);
        printf("\nAlphabets:");
        for(i=0;i<noalpha;i++)
                  printf("%c ",alphabet[i]);
        printf("\nStates :" );
        for(i=1;i<=nostate;i++)
                  print_e_closure(i);

        printf("\nTransitions are...:\n");

        for(i=1;i<=nostate;i++)
        {

                  for(j=0;j<noalpha-1;j++)
                 {
                          for(m=1;m<=nostate;m++)
                                          set[m]=0;
                          for(k=0;e_closure[i][k]!=0;k++)
                          {

                                    t=e_closure[i][k];
                                   temp=transition[t][j];
                                   while(temp!=NULL)
                                  {

                                            unionclosure(temp->st);
                                           temp=temp->link;
                                   }
                           }
                         printf("\n");
                         print_e_closure(i);
                         printf("%c\t",alphabet[j]   );
                         printf("{");
                         for(n=1;n<=nostate;n++)
                         {
                                     if(set[n]!=0)
                                             printf("q%d,",n);
                         }
                          printf("}");
                }
        }
        printf("\nFinal states:");
        findfinalstate();



}

void findclosure(int x,int sta)
{
            struct node *temp;
            int i;
           if(buffer[x])
```

```
                    return;
             e_closure[sta][c++]=x;
            buffer[x]=1;
             if(alphabet[noalpha-1]=='e' && transition[x][noalpha-1]!=NULL)
                {
                            temp=transition[x][noalpha-1];
                            while(temp!=NULL)
                           {
                                       findclosure(temp->st,sta);
                                       temp=temp->link;
                            }
                }
  }

void insert_trantbl(int r,char c,int s)
{
           int j;
           struct node *temp;
            j=findalpha(c);
          if(j==999)
          {
                     printf("error\n");
                    exit(0);
           }
         temp=(struct node *) malloc(sizeof(struct node));
         temp->st=s;
         temp->link=transition[r][j];
         transition[r][j]=temp;
}

int findalpha(char c)
{
            int i;
            for(i=0;i<noalpha;i++)
                   if(alphabet[i]==c)
                          return i;

                return(999);


}

void unionclosure(int i)
{
              int j=0,k;
             while(e_closure[i][j]!=0)
             {
                      k=e_closure[i][j];
                      set[k]=1;
                      j++;
             }
}
void findfinalstate()
{
            int i,j,k,t;
            for(i=0;i<nofinal;i++)
           {
                     for(j=1;j<=nostate;j++)
                     {
                             for(k=0;e_closure[j][k]!=0;k++)
                               {
                                        if(e_closure[j][k]==finalstate[i])
                                       {

                                                print_e_closure(j);
                                       }
```

```
                                }
                        }
                }


  }

void print_e_closure(int i)
{
        int j=0;
        printf("{");
       if(e_closure[i][j]!=0)
                        printf("q%d,",e_closure[i][0]);
         printf("}\t");
}
```

```
^Z
[1]+  Stopped                 ./a.out
akhil@Ubuntu:~/Compiler-Lab/5)nfa with e to nfa without e$ ./a.out
enter the number of alphabets?
3
NOTE:- [ use letter e as epsilon]
NOTE:- [e must be last character ,if it is present]

Enter alphabets?
a
b
e
Enter the number of states?
5
Enter the start state?
1
Enter the number of final states?
1
Enter the final states?
3
Enter no of transition?
7
NOTE:- [Transition is in the form--> qno   alphabet   qno]
NOTE:- [States number must be greater than zero]

Enter transition?
1a2
1e3
2a1
3b4
4b3
3a5
5b3

Equivalent NFA without epsilon
---------------------------------
start state:{q1,}
Alphabets:a b e
States :{q1,}   {q2,}   {q3,}   {q4,}   {q5,}
Transitions are...:

{q1,}   a       {q2,q5,}
{q1,}   b       {q4,}
{q2,}   a       {q1,q3,}
{q2,}   b       {}
{q3,}   a       {q5,}
{q3,}   b       {q4,}
{q4,}   a       {}
{q4,}   b       {q3,}
{q5,}   a       {}
{q5,}   b       {q3,}
akhil@Ubuntu:~/Compiler-Lab/5)nfa with e to nfa without e$
```