

```

#include<stdio.h>
#include<stdlib.h>
struct node {
    int st;
    struct node *link;
};
struct node1 {
    int nst[20];
};

void insert(int ,char, int);
int findalpha(char);
void findfinalstate(void);
int insertdfastate(struct node1);
int compare(struct node1,struct node1);
void printnewstate(struct node1);
static int
set[20],nostate,noalpha,s,notransition,nofinal,start,finalstate[20],c,r,buffer[20];
int complete=-1;
char alphabet[20];
static int eclosure[20][20]={0};
struct node1 hash[20];
struct node * transition[20][20]={NULL};
void main() {
    int i,j,k,m,t,n,l;
    struct node *temp;
    struct node1 newstate={0},tmpstate={0};

    printf("Enter the number of alphabets?\n");
    printf("NOTE:- [ use letter e as epsilon]\n");
    printf("NOTE:- [e must be last character ,if it is present]\n");
    printf("\nEnter No of alphabets and alphabets?\n");
    scanf("%d",&noalpha);
    getchar();
    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(r,c,s);
    }
    for(i=0;i<20;i++) {

```

```

        for(j=0;j<20;j++) hash[i].nst[j]=0;
    }
    complete=-1;
    i=-1;
    printf("\nEquivalent DFA.....\n");
    printf(".....\n");
    printf("Transitions of DFA\n");
    newstate.nst[start]=start;
    insertdfastate(newstate);
    while(i!=complete) {
        i++;
        newstate=hash[i];
        for(k=0;k<noalpha;k++) {
            c=0;
            for(j=1;j<=nostate;j++) set[j]=0;
            for(j=1;j<=nostate;j++) {
                l=newstate.nst[j];
                if(l!=0) {
                    temp=transition[l][k];
                    while(temp!=NULL) {
                        if(set[temp->st]==0) {
                            c++;
                            set[temp->st]=temp->st;
                        }
                        temp=temp->link;
                    }
                }
            }
            printf("\n");
            if(c!=0) {
                for(m=1;m<=nostate;m++)
                    tmpstate.nst[m]=set[m];

                insertdfastate(tmpstate);
                printnewstate(newstate);
                printf("%c\t", alphabet[k]);
                printnewstate(tmpstate);
                printf("\n");
            } else {
                printnewstate(newstate);
                printf("%c\t", alphabet[k]);
                printf("NULL\n");
            }
        }
    }
    printf("\nStates of DFA:\n");
    for(i=0;i<=complete;i++)
        printnewstate(hash[i]);
    printf("\n Alphabets:\n");
    for(i=0;i<noalpha;i++)
        printf("%c\t", alphabet[i]);
    printf("\n Start State:\n");
    printf("q%d", start);
    printf("\nFinal states:\n");
    findfinalstate();
}

int insertdfastate(struct node1 newstate) {
    int i;
    for(i=0;i<=complete;i++) {

```

```

        if(compare(hash[i],newstate))
            return 0;
    }
    complete++;
    hash[complete]=newstate;
    return 1;
}

int compare(struct node1 a,struct node1 b) {
    int i;
    for(i=1;i<=nostate;i++) {
        if(a.nst[i]!=b.nst[i])
            return 0;
    }
    return 1;
}

void insert(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 findfinalstate() {
    int i,j,k,t;

    for(i=0;i<=complete;i++) {
        for(j=1;j<=nostate;j++) {
            for(k=0;k<nofinal;k++) {
                if(hash[i].nst[j]==finalstate[k]) {
                    printnewstate(hash[i]);
                    printf("\t");
                    j=nostate;
                    break;
                }
            }
        }
    }
}

```

```
void printnewstate(struct node1 state) {  
    int j;  
    printf("{");  
    for(j=1;j<=nostate;j++) {  
        if(state.nst[j]!=0)  
            printf("q%d,", state.nst[j]);  
    }  
    printf("}\t");  
}
```