EXPT 1

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<ctype.h>

int Keyword(char buffer[]){
char keywords[32][10] = {"auto","break","case","char","const","continue","default",
"do","double","else","enum","extern","float","for","goto",
"if","int","long","register","return","short","signed",
"sizeof","static","struct","switch","typedef","union",
"unsigned","void","volatile","while"};
int i, flag = 0;
for(i = 0; i < 32; ++i){
if(strcmp(keywords[i], buffer) == 0){
flag = 1;
break;
}
}
return flag;
}

int main(){
char ch, buffer[15], operators[] = "+-*/%=";
FILE *fp;
int i,j=0;
fp = fopen("program.txt","r");
if(fp == NULL){
printf("error while opening the
file\n");exit(0);
}
while((ch = fgetc(fp)) != EOF){
   for(i = 0; i < 6; ++i){
   if(ch == operators[i])
   printf("%c - operator\n", ch);
   }

   if(isalnum(ch)){ buffer[j++]
   = ch;
   }
   else if((ch == ' ' || ch == '\n') && (j != 0)){
   buffer[j] = '\0';
   j = 0;

   if(Keyword(buffer) == 1) printf("%s
   - keyword\n", buffer);else
   printf("%s - indentifier\n", buffer);
   }

}
fclose(fp);
return 0;
```

}

OUTPUT

```
student@P29:~$ cd Desktop
student@P29:~/Desktop$ gedit lex.c
student@P29:~/Desktop$ gcc lex.c
student@P29:~/Desktop$ ./a.out
void - keyword
main - indentifier
int - keyword
abc - indentifier
= - operator
+ - operator
cab - indentifier
```

EXPT 2

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

char result[20][20],copy[3],states[20][20];void
add_state(char a[3],int i){
        strcpy(result[i],a);
}

void display(int n){
        int k=0;
        printf("\n Epsilon closure of %s = { ",copy);while(k < n){
                printf(" %s",result[k]);k++;
        }
        printf(" } \n");
}

int main(){
    FILE *INPUT;
    INPUT=fopen("input.dat","r");char
    state[3];
int end,i=0,n,k=0;
char state1[3],input[3],state2[3]; printf("\n Enter
the no of states: ");scanf("%d",&n);
printf("\n Enter the states \n");
for(k=0;k<3;k++){
            scanf("%s",states[k]);

    }
    for( k=0;k<n;k++){i=0;
    strcpy(state,states[k]);
    strcpy(copy,state);
```

```
                add_state(state,i++);
                while(1){
                        end = fscanf(INPUT,"%s%s%s",state1,input,state2);if (end ==
                        EOF ){
                                break;
}

                        if( strcmp(state,state1) == 0 ){
                                if( strcmp(input,"e") == 0 ) {
                                        add_state(state2,i++);
                                        strcpy(state, state2);
                                 }
                        }
}

display(i);
rewind(INPUT);
}
return 0;
}
```

OUTPUT

```
student@P32:~$ cd Desktop
student@P32:~/Desktop$ gedit cf.c
student@P32:~/Desktop$ gcc cf.c
student@P32:~/Desktop$ ./a.out

 Enter the no of states: 3

 Enter the states
q0
q1
q2

 Epsilon closure of q0 = {  q0 q1 q2 }

 Epsilon closure of q1 = {  q1 q2 }

 Epsilon closure of q2 = {  q2 }
student@P32:~/Desktop$
```

NFA TO DFA

```
#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("Trnsitions 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");
}
```

```
student@P33:~$ cd Desktop
student@P33:~/Desktop$ gcc nfa.c
nfa.c: In function ‘main’:
nfa.c:57:9: warning: too many arguments for format [-Wformat-extra-a
   57 |  printf("NOTE:- [Transition is in the form-> qno alphabet qr
      |         ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
nfa.c:66:14: warning: format ‘%c’ expects argument of type ‘char *’,
   66 |   scanf("%d %c%d",&r,&c,&s);
      |             ~^       ~~
      |              |       |
      |              char *  int *
      |             %lc
student@P33:~/Desktop$ ./a.out
Enter the number of alphabets?
NOTE:- [ use letter e as epsilon]
NOTE:- [e must be last character ,if it is present]

Enter No of alphabets and alphabets?
2
a
b
Enter the number of states?
4
Enter the start state?
1
Enter the number of final states?
2
Enter the final states?

3
4
Enter no of transition?
8
NOTE:- [Transition is in the form-> qno alphabet qno]
NOTE:- [States number must be greater than zero]

Enter transition?
 1 a 1
```

```
Enter the final states?

3
4
Enter no of transition?
8
NOTE:- [Transition is in the form-> qno alphabet qno]
NOTE:- [States number must be greater than zero]

Enter transition?
 1 a 1
1 b 1
1 a 2
2 b 2
2 a 3
3 a 4
3 b 4
4 b 3

Equivalent DFA.....
Trnsitions of DFA

{q1,}   a       {q1,q2,}

{q1,}   b       {q1,}

{q1,q2,}        a       {q1,q2,q3,}

{q1,q2,}        b       {q1,q2,}

{q1,q2,q3,}     a       {q1,q2,q3,q4,}

{q1,q2,q3,}     b       {q1,q2,q4,}

{q1,q2,q3,q4,}  a       {q1,q2,q3,q4,}

{q1,q2,q3,q4,}  b       {q1,q2,q3,q4,}
```

```
{q1,q2,q3,}     b       {q1,q2,q4,}
{q1,q2,q3,q4,}  a       {q1,q2,q3,q4,}
{q1,q2,q3,q4,}  b       {q1,q2,q3,q4,}
{q1,q2,q4,}     a       {q1,q2,q3,}
{q1,q2,q4,}     b       {q1,q2,q3,}
States of DFA:
{q1,}   {q1,q2,}        {q1,q2,q3,}     {q1,q2,q3,q4,}  {q1,q2,q4
 Alphabets:
a       b
 Start State:
q1
Final states:
{q1,q2,q3,}             {q1,q2,q3,q4,}          {q1,q2,q4,}
```

EXPT 4

```
#include<stdio.h>
#include<math.h>
#include<string.h>
#include<ctype.h>
#include<stdlib.h>
int n,m=0,p,i=0,j=0;
char a[10][10],f[10];
void follow(char c);
void first(char  c);
int main(){

        int i,z;
        char c,ch;
        //clrscr();
        printf("Enter the no of
        prooductions:\n");scanf("%d",&n);
        printf("Enter the
        productions:\n");for(i=0;i<n;i++)
                scanf("%s%c",a[i],&ch);
        do{
                m=0;
                printf("Enter the elemets whose fisrt & follow is to be
        found:");scanf("%c",&c);
        first(c);
        printf("First(%c)={",c);
        for(i=0;i<m;i++)
        printf("%c",f[i]);
        printf("}\n");
        strcpy(f," ");
        //flushall();
        m=0;
        follow(c);
        printf("Follow(%c)={",c);
        for(i=0;i<m;i++)
        printf("%c",f[i]);
        printf("}\n");
                printf("Continue(0/1)?");
                scanf("%d%c",&z,&ch);
        }while(z==1);
        return(0);
}
void first(char c)
{
        int k;
        if(!isupper(c))
        f[m++]=c;
        for(k=0;k<n;k++)
        {
```

```
                if(a[k][0]==c)
                {
                        if(a[k][2]=='$')
                        follow(a[k][0]);
                else if(islower(a[k][2]))
                        f[m++]=a[k][2];
                else first(a[k][2]);
                }
        }
}
void follow(char c)
{
        if(a[0][0]==c)
                f[m++]='$';
        for(i=0;i<n;i++)
        {
                for(j=2;j<strlen(a[i]);j++)
                {
                        if(a[i][j]==c)
                        {
                                if(a[i][j+1]!='\0')
                                        first(a[i][j+1]);
                                if(a[i][j+1]=='\0' && c!=a[i][0])
                                        follow(a[i][0]);
                        }
                }
        }
```

OUTPUT

```
student@P32:~/Desktop$ ./a.out
Enter the no of prooductions:
5
Enter the productions:
S=AbCd
A=Cf
A=a
C=Gg
E=h
Enter the elemets whose fisrt & follow is to be found:S
First(S)={a}
Follow(S)={$}
Continue(0/1)?1
Enter the elemets whose fisrt & follow is to be found:A
First(A)={a}
Follow(A)={b}
Continue(0/1)?
```