Automata

Question One

Translating RE into an NFA

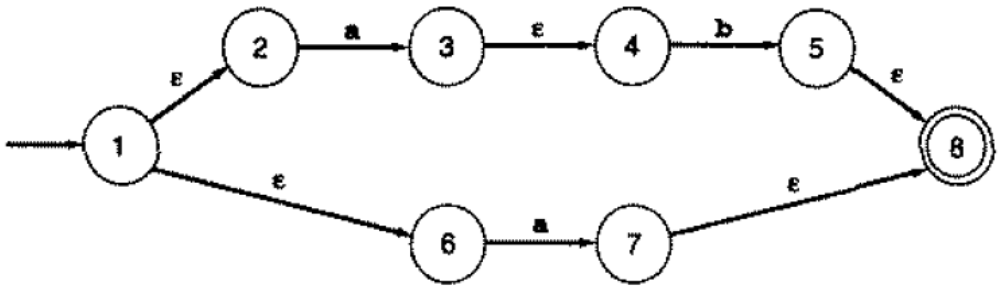
(Use ϵ–transitions to "glue together" the machines of each piece of the regular expression)

1. **(a | b | ab)**
2. **((ε|a)b*)***
3. **(a|b)* ac**

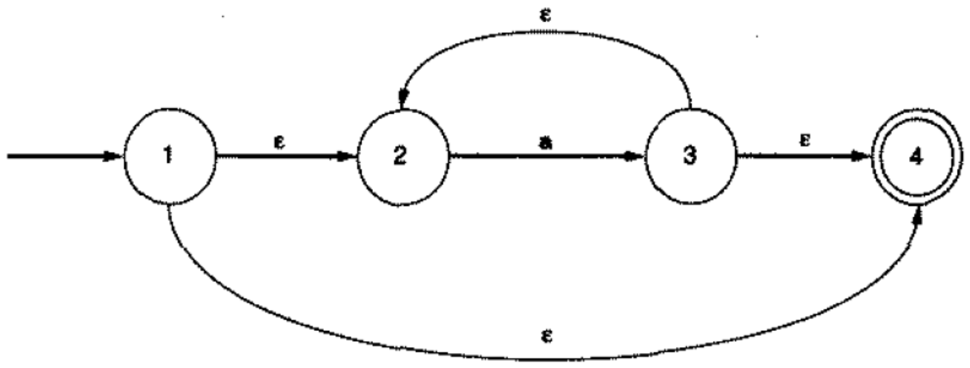
Question Two

Build an equivalent DFA for the following NFA using subset construction

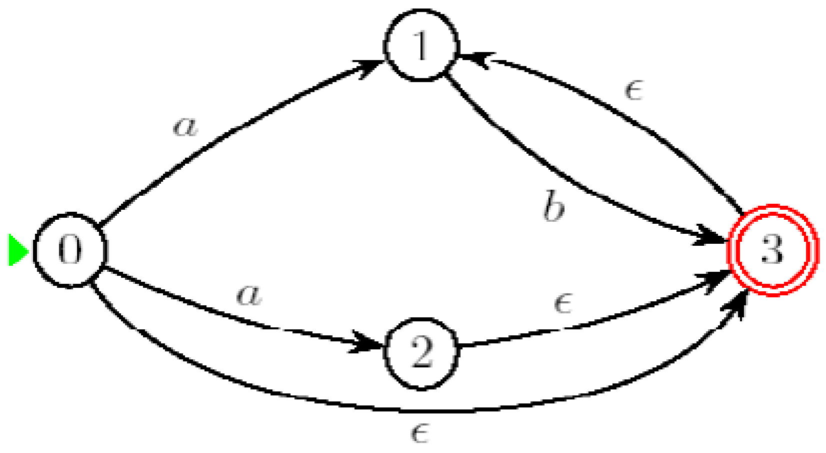
1.



2.



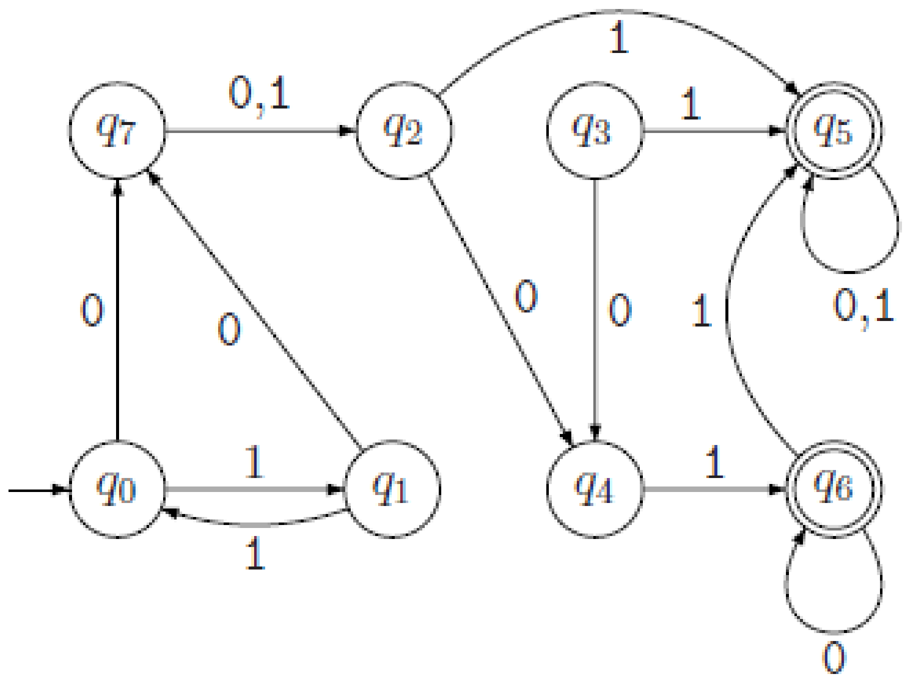
3.



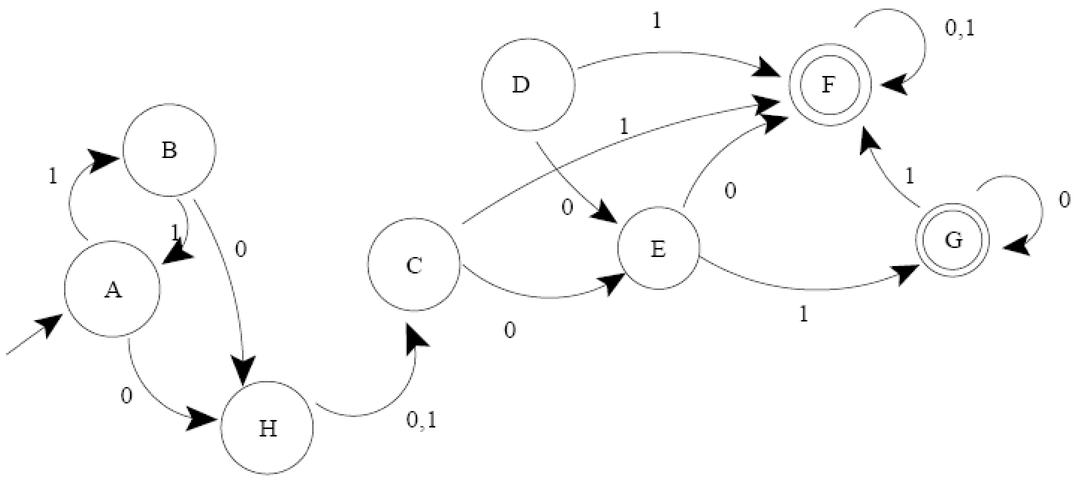
Question Three

Minimizing DFA

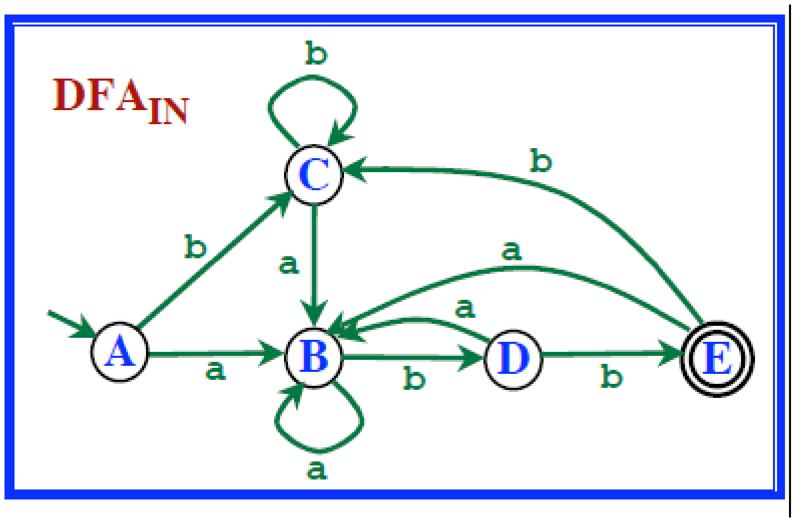
1.



2.



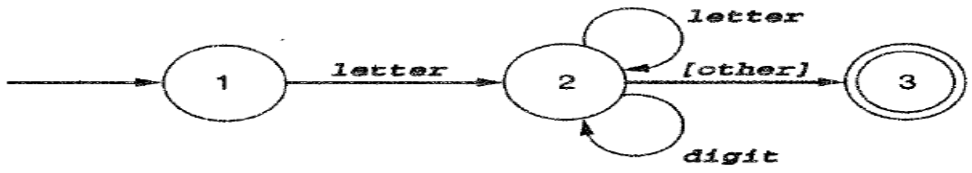
3.



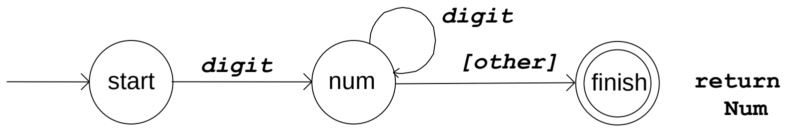
Question Four

Translate the following DFA into Code

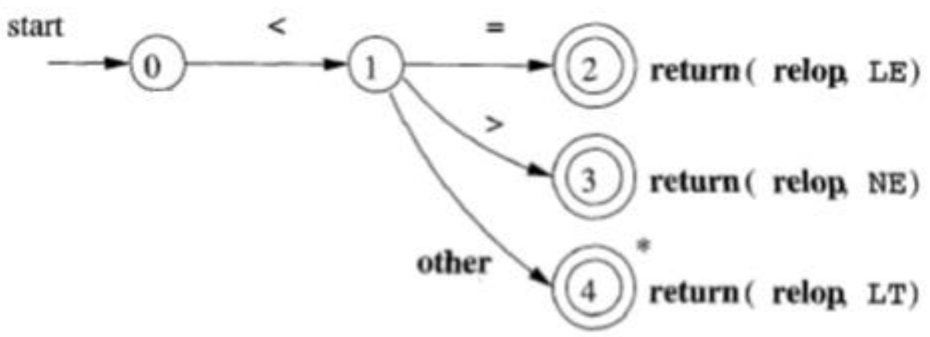
1.



2.



3.



Question Five

Use of FLEX:

1. **Write FLEX input file to print only line that end or begin with the letter a.**
2. **A lexer print out all HTML tags (start with < and end with>)**
3. **Write FLEX input file to print the total number of lines that begin with uppercase Letter.**

4. What is the output of the following flex scanners

```
%%
a*b       {printf("<%s ,%s>","1",yytext);}
(a|b)*b {printf("<%s ,%s>","2",yytext);}
c*        {printf("<%s ,%s>","3",yytext);}
%%
int yywrap(){return 1;}
int main(){
yylex();
return 0;
}
```

Input: **aaabccabbb**

5. What is the output of the following flex scanners

```
%%
aa*          {printf("<%s ,%s>","1",yytext);}
c(a|b)*      {printf("<%s ,%s>","2",yytext);}
ab*c         {printf("<%s ,%s>","3",yytext);}
```

```
caa*            {printf("<%s ,%s>","4",yytext);}
b*aa*(c|e)    {printf("<%s ,%s>","5",yytext);}

%%
int yywrap(){return 1;}
int main(){
yylex();
return 0;
}
```

Input :aaabccabbb

Which rule cannot be matched

Question Six

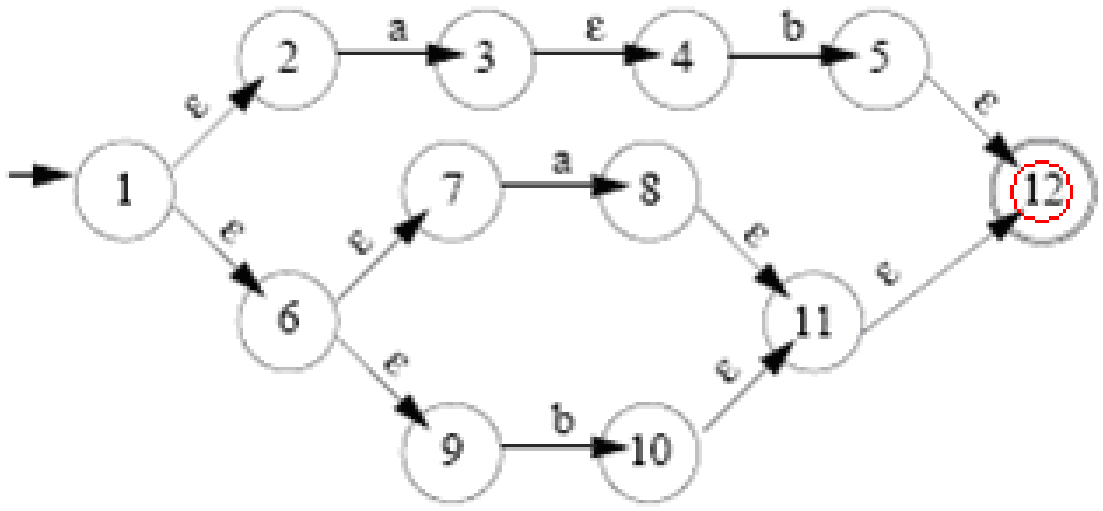
For RE→ letter(letter|digit)*

a) **Convert RE to NFA**
b) **convert NFA to DFA**
c) **Minimize DFA**

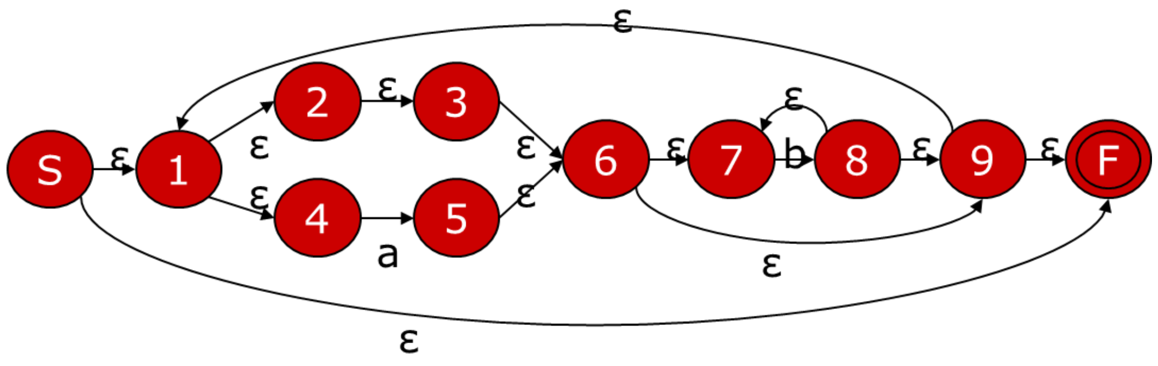
Solutions

Question One

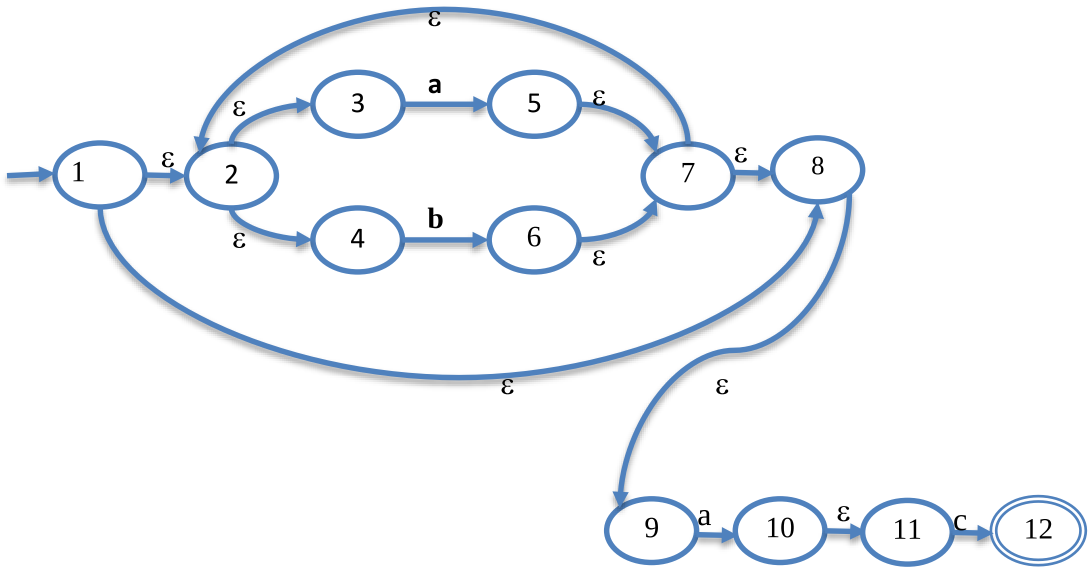
1. (a | b | ab)



2. ((ε|a)b*)*

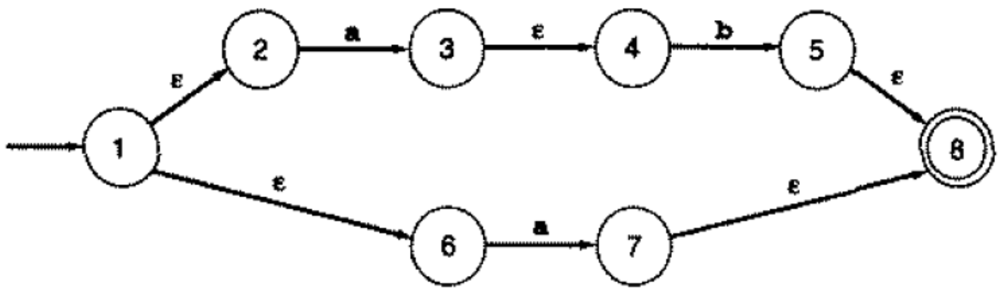


3. (a|b)* ac



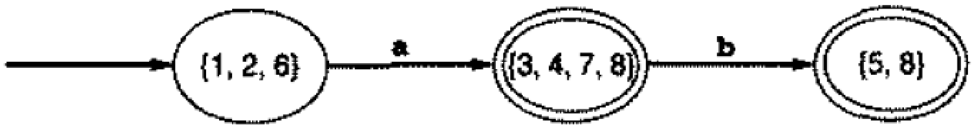
Question Two

1.

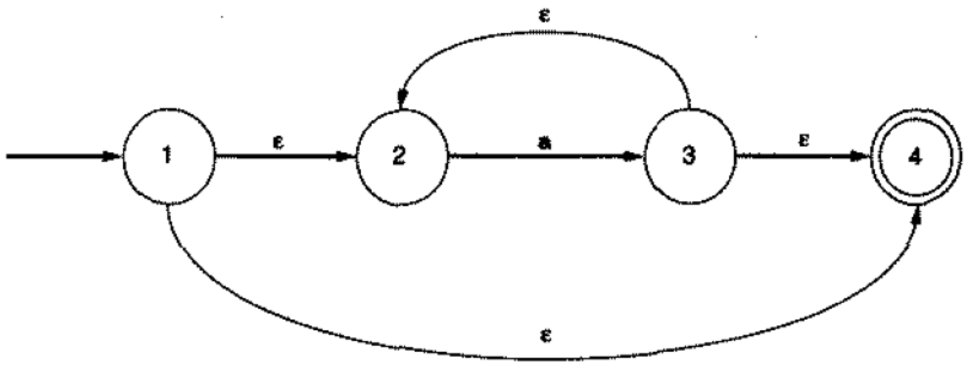


Solution

NFA	a	b
{1,2,6}	{3,4,7,8}	
{3,4,7,8}		{5,8}

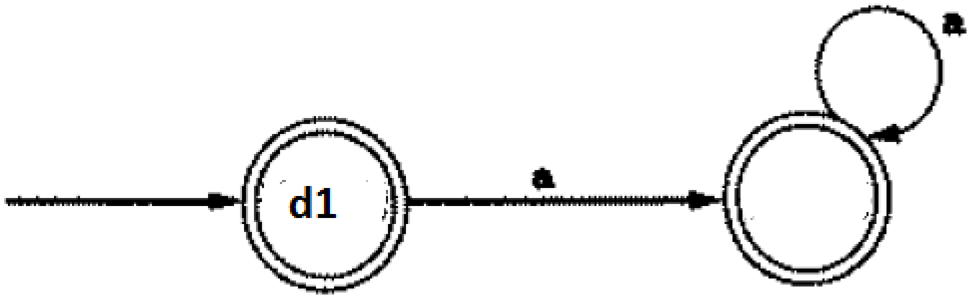


2.

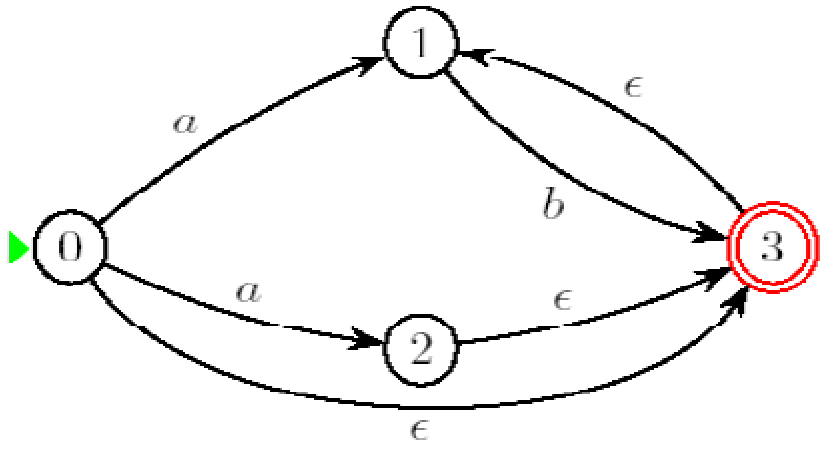


Solution

NFA State	a
d1={1,2,4}	d2={2,3,4}
d2={2,3,4}	d2={2,3,4}

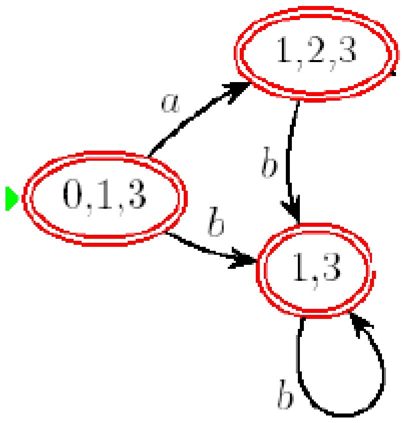


3.



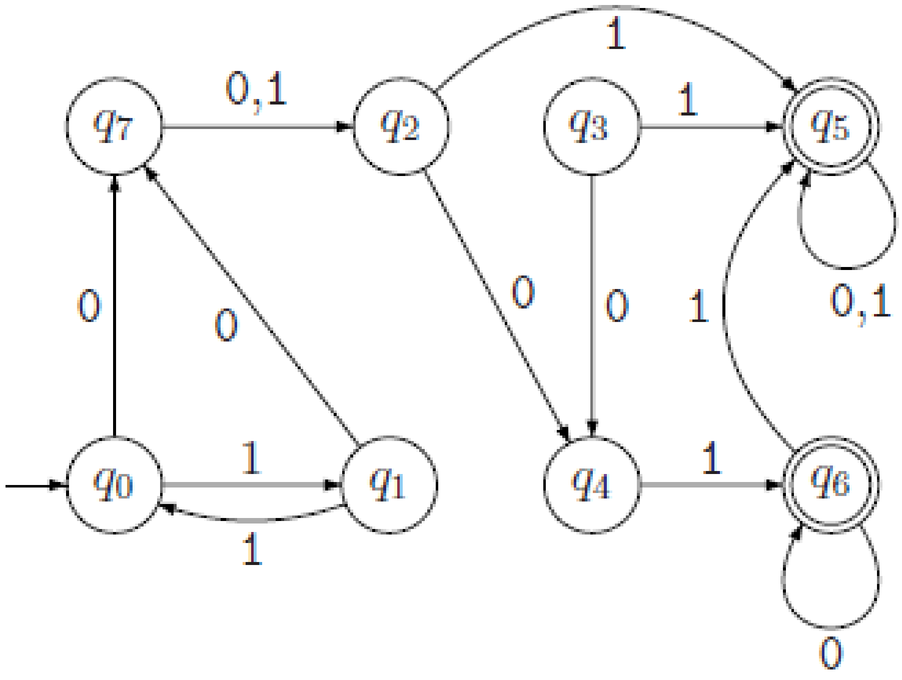
Solution

nfa	**a**	**b**
{0,1,3}	**{1,2,3}**	**{1,3}**
{1,2,3}		**{1,3}**
{1,3}		**{1,3}**

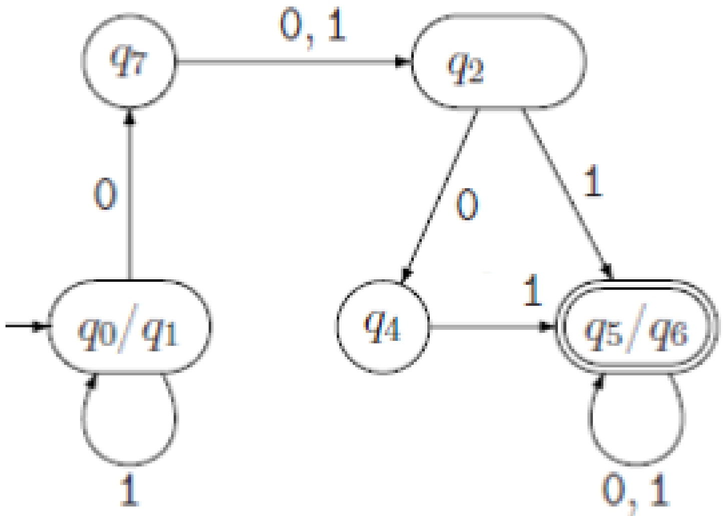


Question Three

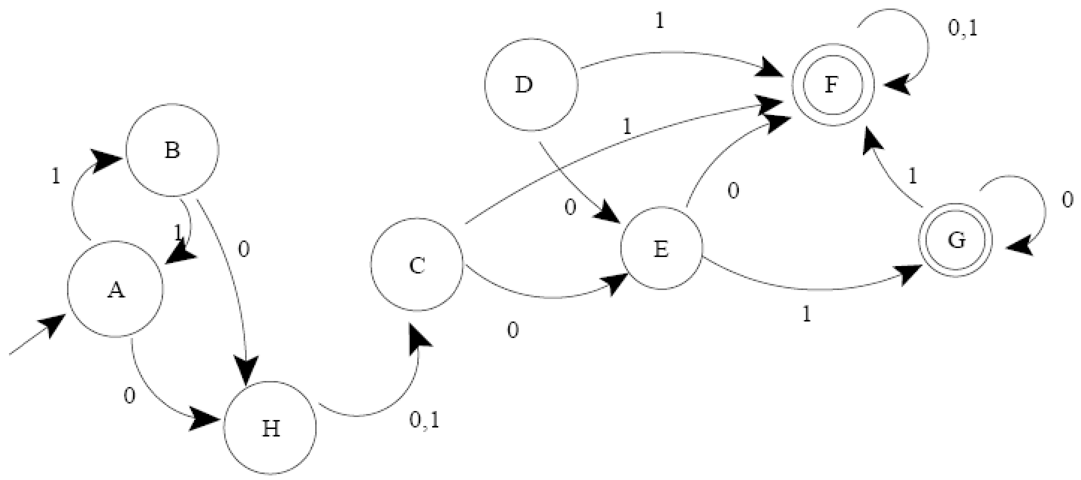
1.



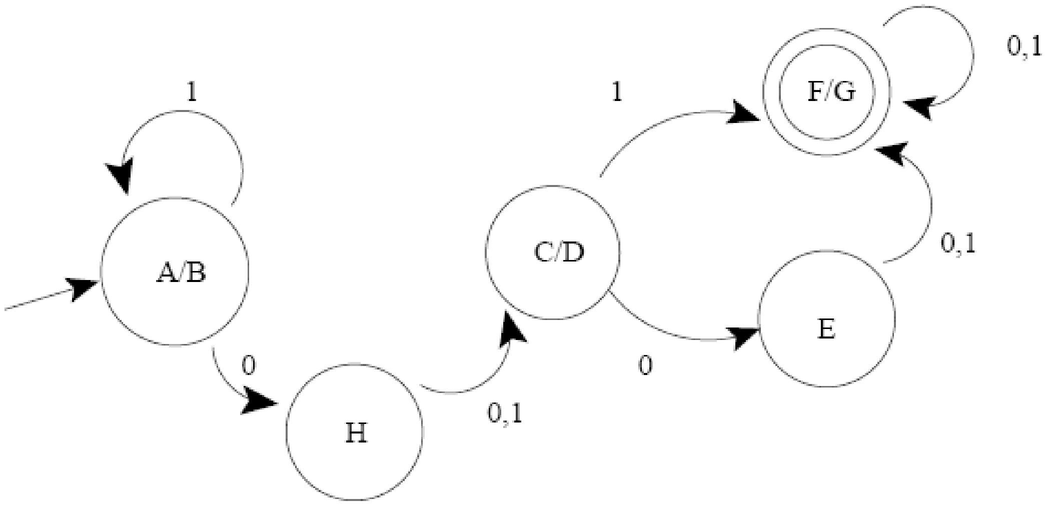
Solution



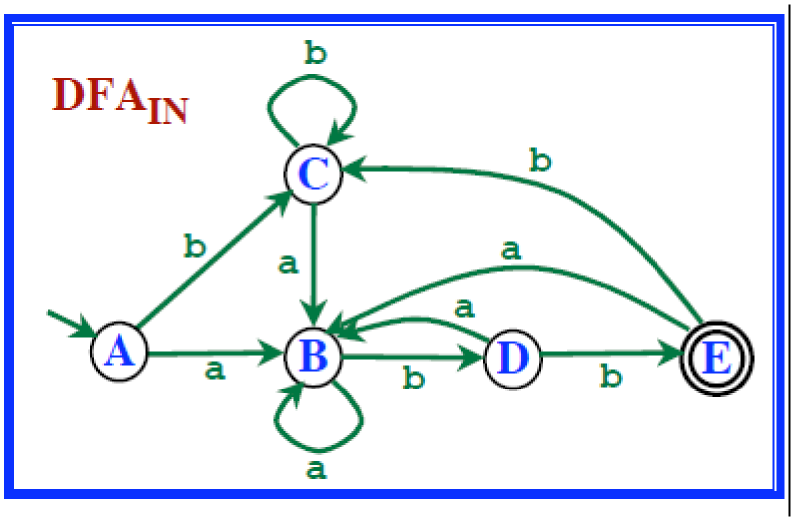
2.

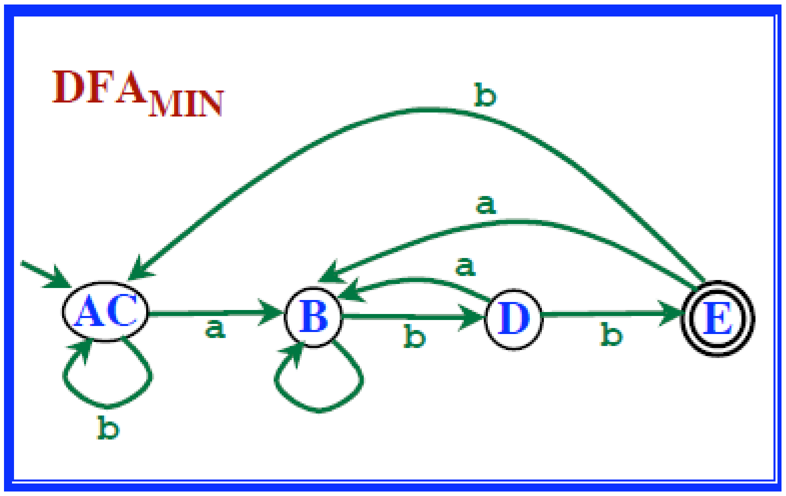


Solution



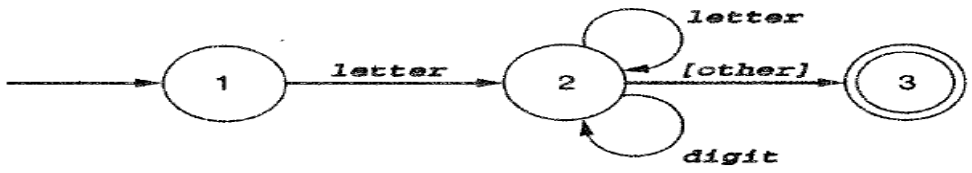
3.



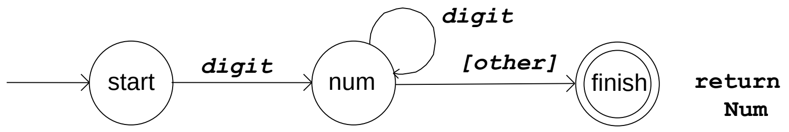


Question Four

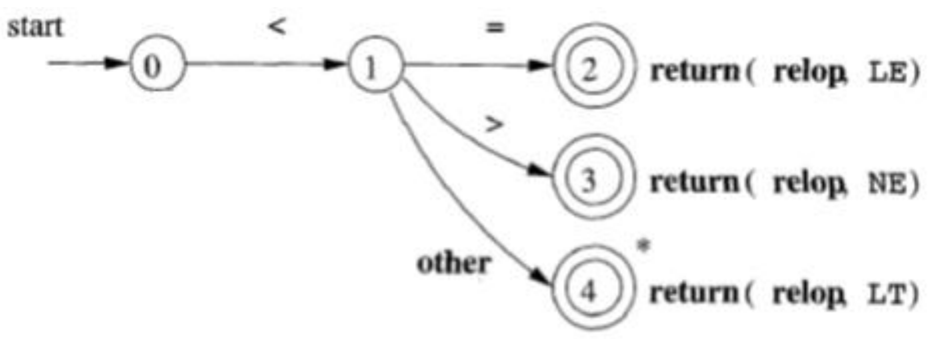
1.



2.



3.



Solution

1.

```
State =1;
Get the input string;
while (state != 3 && state != error)
{
 switch (state) {
      case 1: if (isalpha(input)) {
                        advance(input);
      state = 2;
      }
              else state = error;
        break;
   case 2: if (isalpha(input)|| isdigit(input))
                  advance(input);
      else
      state = 3;
      break;
   default: break;
 }
if (state == 3) return ID;
  else return ERROR;
```

2.

```
state = start;
advance(input);
while (state != finish && state != error)
 switch (state) {
   case start: if (isdigit(input)) {
                        advance(input); state = num;}
                        else state = error; break;
   case num: if (!isdigit(input))
```

```
                                            state = finish;
                                            else advance(input); break;
        default: break;
       }
     if (state == finish) return ID;
       else return ERROR;
```

Question Five:

Use of FLEX:

1. **Write FLEX input file to print only line that end or begin with the letter a.**

```
begin_end_a  (a.*\n|.*a\n)
other           .*\n
%%
{begin_end_a}       {printf ("%s",yytext);}
{other}

%%
int main(int argc, char *argv[]){
printf("Enter your Code :\n");
yylex();
system("PAUSE");
return 0;
}
```

2. **A lexer print out all HTML tags (start with < and end with>)**

```
HtmlTag       <.*>

%%

{HtmlTag}    {printf ("%s",(yytext));}

.*

%%

int yywrap(){
```

```
return 1;

}

int main(int argc, char *argv[]){

printf("Enter your Code :\n");



yylex();

system("PAUSE");

return 0;

}
```

3. **Write FLEX input file to print the total number of lines that begin with uppercase Letter.**

```
%{
int upperCount=0;
%}
Upper[A-Z].*\n
%%
{Upper}       {upperCount++;}
.*
%%
int main(int argc, char *argv[]){
printf("Enter your Code :\n");
yylex();
printf("upperCount %d",upperCount);
system("PAUSE");
return 0;
    }
```

4. What is the output of the following flex scanners

```
%%
a*b       {printf("<%s ,%s>","1",yytext);}
(a|b)*b {printf("<%s ,%s>","2",yytext);}
c*        {printf("<%s ,%s>","3",yytext);}
%%
int yywrap(){return 1;}
int main(){
yylex();
return 0;
}
```

Input: **aaabccabbb**

```
<1 ,aaab><3 ,cc><2 ,abbb>
```

5. What is the output of the following flex scanners

```
%%
aa*          {printf("<%s ,%s>","1",yytext);}
c(a|b)*      {printf("<%s ,%s>","2",yytext);}
ab*c         {printf("<%s ,%s>","3",yytext);}
caa*         {printf("<%s ,%s>","4",yytext);}
b*aa*(c|e)   {printf("<%s ,%s>","5",yytext);}

%%
int yywrap(){return 1;}
int main(){
yylex();
return 0;
}
```

Input :aaabccabbb

```
<1 ,aaa>b<2 ,c><2 ,cabbb>
```

Which rule cannot be matched

Rule4 caa*

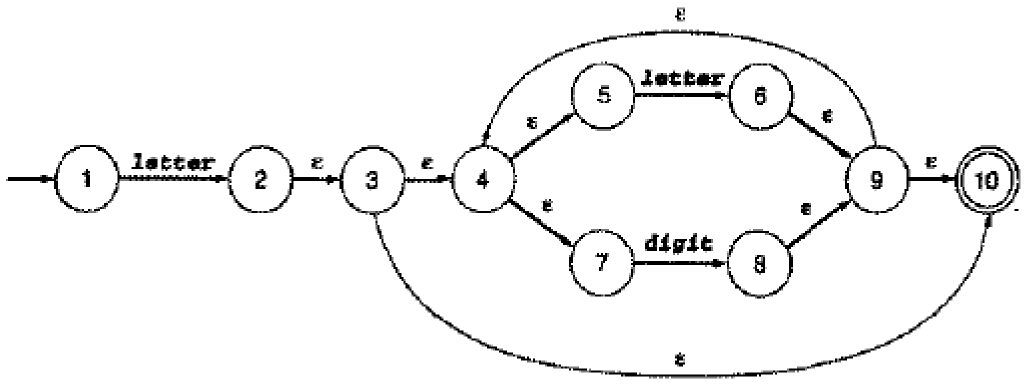
Question Six

For RE→ letter(letter|digit)*

d) Convert RE to NFA
e) convert NFA to DFA
f) Minimize DFA

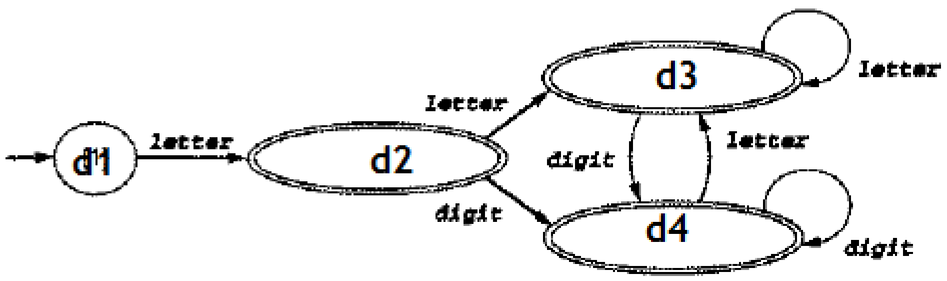
Solution

a) Convert RE to NFA

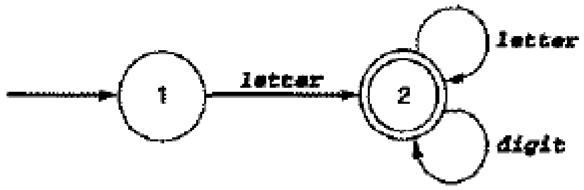


b) convert NFA to DFA

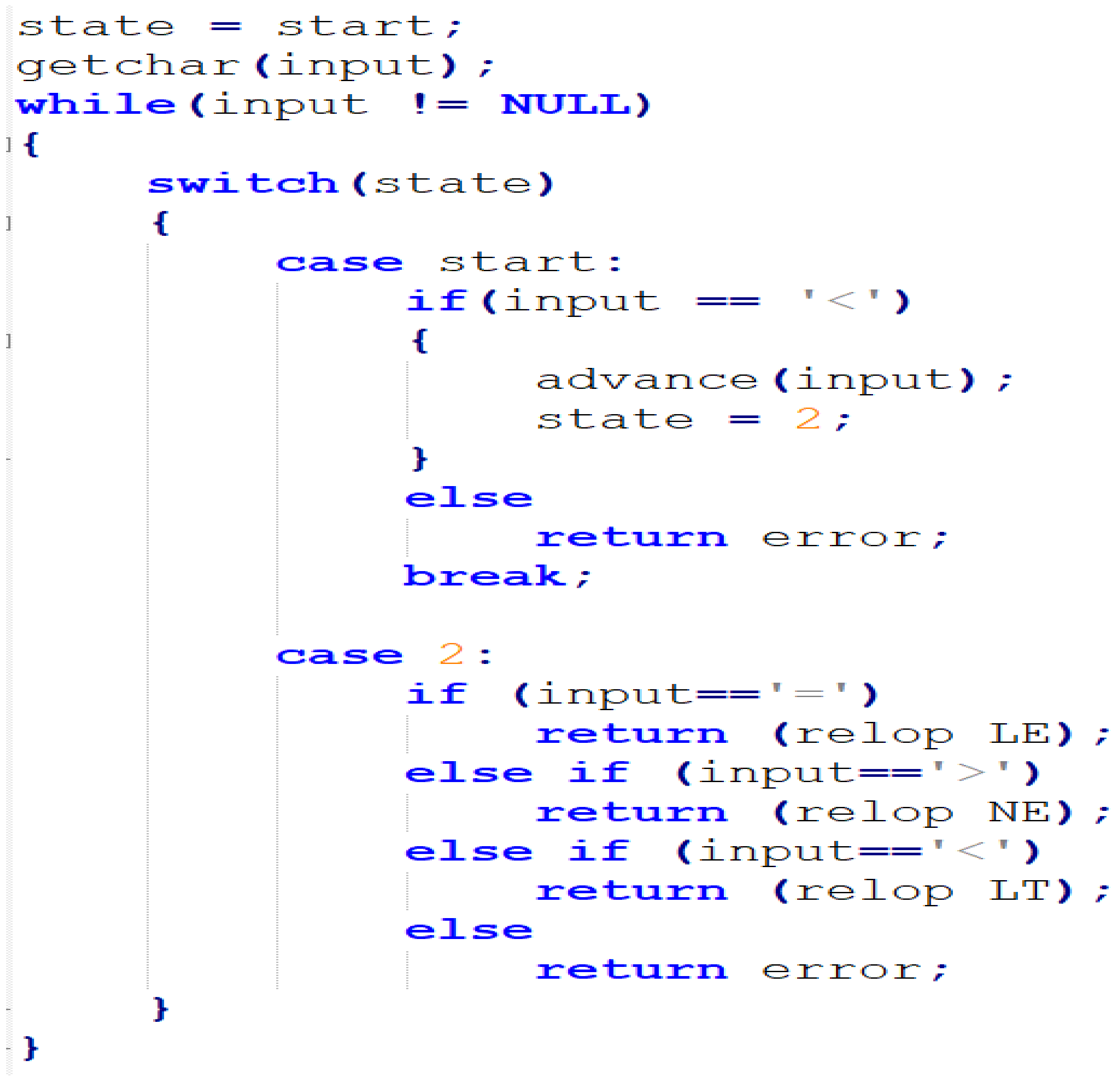
NFA State	Letter	Digit
d1=n1	d2={2,3,4,5,7,10}	none
d2={2,3,4,5,7,10}	d3={4,5,6,7,9,10}	d4={4,5,7,8,9,10}
d3={4,5,6,7,9,10}	d3={4,5,6,7,9,10}	d4={4,5,7,8,9,10}
d4={4,5,7,8,9,10}	d3={4,5,6,7,9,10}	d4={4,5,7,8,9,10}



c) Minimize DFA



3.



```
state = start;
getchar(input);
while(input != NULL)
{
      switch(state)
      {
            case start:
                  if(input == '<')
                  {
                        advance(input);
                        state = 2;
                  }
                  else
                        return error;
                  break;

            case 2:
                  if (input=='=')
                        return (relop LE);
                  else if (input=='>')
                        return (relop NE);
                  else if (input=='<')
                        return (relop LT);
                  else
                        return error;
      }
}
```