**Problem Statement 1: Write a C Program to scan and count the number of characters, words, and lines in a given text file.**

**Input from keyboard**: file name

**Output on the screen:**

Number of characters:

Number of words:

Number of lines:

**Description of the problem statement**: we will input the full path of a text file on console and after that it displays numbers of characters, words, and lines of that file.

**Algorithm:**

**Start**

1. Input the full path of file or input filename
2. Open that file in read mode and print an error message if doesn’t exist.
3. Initialize characters=words=lines=0
4. Read character one by one from the file and repeat these steps

Characters++

if(ch==' '|| ch=='\n') then

words++;

characters--; end if

if(ch=='\n')then

lines++; end if

End loop

1. if(characters>0) then

lines++; end if

1. print numbers of characters, words, and lines

**Stop**

**C code:**

#include <stdio.h>

void main()

{

FILE \*fptr;

int words,lines,characters;

words=characters=0;

lines=0;

char path[100];

printf("enter path/filename:");

scanf("%s",path);

fptr = fopen(path, "r");

if (fptr!=NULL)

{

int ch;

while( (ch=fgetc(fptr)) !=EOF)

{

characters++;

if(ch==' '|| ch=='\n')

{

words++;

characters--;

}

if(ch=='\n')

{

lines++;

}

}

if(characters>0)

{

lines++;

words++;

}

}

else

printf("\nFile Doesnot exist");

fclose(fptr);

printf("\nNumber of characters:%d\n",characters);

printf("Number of words:%d\n",words);

printf("Number of lines:%d\n",lines);

}

**Output**

Text

Description automatically generated

**File.txt**

Text

Description automatically generated with medium confidence

**PROBLEM STATEMENT 2: Write a C Program to implement a given**

**Deterministic Finite Automaton (DFA) that accepts/rejects a given input string.**

**Input from Keyboard:** DFA parameters, and Input String

**Output on the Screen:** 5-tuple representation of given DFA, and acceptance/rejection status of the input string.

**Description of the problem statement:**

Objective is to implement a given Deterministic Finite Automaton (DFA) that accepts/rejects a given input string.

**Algorithm:**

**START**

1. READ: States, Input\_Alphabet, final\_states, Intial\_state, Transition table

2. READ input\_string (string)

3. CALL logic FUNCTION

INITIALIZE dfa = -1

WHILE (\*string != '\0') DO

IF(dfa == -1) THEN

dfa = 0; END IF

FOR j = 0; j < size of Input Alphabet; j++ DO

IF (\*string == Input\_Alphabet [j]) THEN

dfa = Transition table[dfa][j]; END IF

END FOR

string++;

END WHILE

return dfa;

5. IF dfa==final\_state

PRINT: “String is ACCEPTED”;

ELSE

PRINT: “String is REJECTED”;

**STOP**

#include <stdio.h>

int nS;

int nIA;

void information(char\* str,char \*inAlpha,int q0,int fstate,int transTable[nS][nIA],int Q[],int nIA,int nS);

int logic(char \*str, char \*inAlpha, int transT[nS][nIA],int nIA);

void main()

{

printf("how many states: ");

scanf("%d",&nS);

int Q[nS];

printf("\nEnter states: ");

for(int i=0;i<nS;i++){

scanf("%d",&Q[i]);

}

printf("\nhow many input Alphabets: ");

scanf("%d",&nIA);

char inputAlpha[nIA+1];

printf("\nenter inputAlphabets:");

scanf("%s",inputAlpha);

int q0;

printf("\nenter Initial state:");

scanf("%d",&q0);

int fstate;

printf("\nenter final state:");

scanf("%d",&fstate);

int transTable[nS][nIA];

printf("\nEnter values in transition table\n");

for(int i=0;i<nS;i++){

for(int j=0;j<nIA;j++){

scanf("%d",&transTable[i][j]);

}

}

char string[100];

printf("enter the string:");

scanf("%s", string);

information(string,inputAlpha,q0,fstate,transTable,Q,nIA,nS);

int dfa = logic(string, inputAlpha, transTable,nIA);

if (dfa == fstate)

{

printf("String is Accepted");

}

else

{

printf("String is rejected");

}

}

int logic(char \*str, char \*inAlpha, int transT[nS][nIA],int nIA)

{

int i = 0;

int dfa = -1;

while (\*str != '\0')

{

if (dfa == -1)

{

dfa = 0;

}

for (int j = 0; j < nIA; j++)

{

if (\*str == inAlpha[j])

{

dfa = transT[dfa][j];

}

}

str++;

}

return dfa;

}

void information(char\* str,char \*inAlpha,int q0,int fstate,int transTable[nS][nIA],int Q[],int nIA,int nS){

printf("\nInitial states:%d",q0);

printf("\nfinal states:%d",fstate);

printf("\nAll states:");

for(int i=0;i<nS;i++)

{

printf("%d ",Q[i]);

}

printf("\nInput Alphabets:");

while(\*inAlpha!='\0')

{

printf("%c",\*inAlpha);

inAlpha++;

}

printf("\nInput String s:");

while(\*str!='\0')

{

printf("%c",\*str);

str++;

}

printf("\ntransition table:\n");

for(int i=0;i<nS;i++){

for(int j=0;j<nIA;j++){

printf("[%d]",transTable[i][j]);

}

printf("\n");

}

}

Diagram

Description automatically generated

ACCEPTED REJECTED

Text

Description automatically generated Text

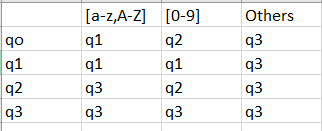
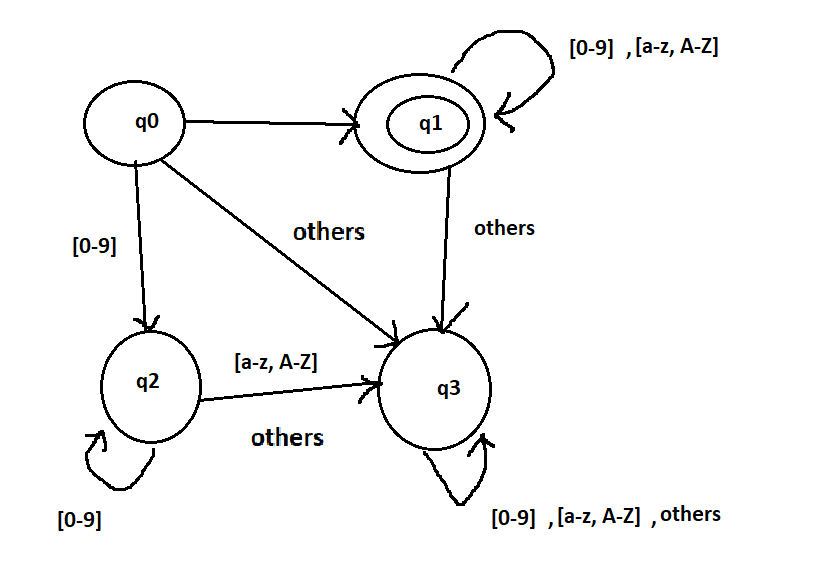
Description automatically generated

**Problem statement 3: Design and implement an automaton in ‘C’ programming language to identify and count substrings that are present in a given input string, as an identifier or number.**

**Input from keyboard:** DFA parameters, and Input String

**Output on the screen:** 5-tuple representation of the designed automaton, and mapping of substrings into identifier or number.

**Description of the problem statement:** Here we are creating a DFA that maps input substring into identifier, number or other. Identifiers are all those strings which start with an alphabet (a-z), followed by any combination of alphabets (a-z) and digits (a-z). Numbers are the strings which are any combination of digits (0-9) only. If the substring is neither identifier nor the number, then it is identified as others



C code

//lab problem 3

#include<stdio.h>

void logic(char \*str,int transT[4][3]);

int isAlphabet(char ch);

int isNumber(char ch);

void information(char\* str,int q0,int fstate,int transTable[4][3],int Q[]);

void main()

{

printf("Enter total states: ");

int Q[4];

for( int i=0; i<4; i++)

{

scanf("%d",&Q[i]);

}

int q0;

printf("\nEnter Initial state: ");

scanf("%d",&q0);

int finalState; // for identifier

printf("\nEnter final state: ");

scanf("%d",&finalState);

fflush(stdin);

char string[100];

printf("\nenter the string that is to be checked: ");

fgets(string,100,stdin);

int transTable[4][3]={{1,2,3},{1,1,3},{3,2,3},{3,3,3}};

information(string,q0,finalState,transTable,Q);

logic(string,transTable);

}

// 0->alpha 1->[0-9] 2->[others]

void logic(char \*str,int transT[4][3]){

char \*start=str;

int i=0;

int inputAlpha;

int dfa=0;

while(str[i]!='\0'){

char ch= str[i];

if(isAlphabet(ch)){

inputAlpha=0;

dfa=transT[dfa][inputAlpha];

}

else if(isNumber(ch)){

inputAlpha=1;

dfa=transT[dfa][inputAlpha];

}

else

{

inputAlpha=2;

dfa=transT[dfa][inputAlpha];

}

i++;

// when a substring has been completely processed

if(str[i]==' ' || str[i]=='\n'){

printf("\n");

while(\*start!=' '&& \*start!='\0' && \*start!='\n' ){

printf("%c",\*start);

start++;

}

if(dfa==1) // q1

printf("\tIdentifier");

else if(dfa==2) //q2

printf("\tNumber");

else //q3

printf("\tOthers");

dfa=0; // initialize back to first state for next substring

start=&str[i+1];// first character of next substring;

i++;

}

}

}

int isAlphabet(char ch){

if((ch>='a' && ch<='z') || (ch>='A' && ch<='Z'))

return 1;

else

return 0;

}

int isNumber(char ch){

if(ch>='0' && ch<='9')

return 1;

else

return 0;

}

void information(char\* str,int q0,int fstate,int transTable[4][3],int Q[]){

printf("\nInitial states:%d",q0);

printf("\nfinal states:%d",fstate);

printf("\nAll states:");

for(int i=0;i<4;i++)

{

printf("%d ",Q[i]);

}

printf("\nInput Alphabets:[A-Z,a-z],[0-9],[other symbols]");

printf("\nInput String s:");

while(\*str!='\0')

{

printf("%c",\*str);

str++;

}

printf("\ntransition table:\n");

for(int i=0;i<4;i++){

for(int j=0;j<3;j++){

printf("[%d]",transTable[i][j]);

}

printf("\n");

}

}

Text

Description automatically generated

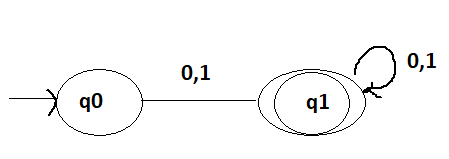
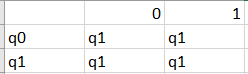
**Problem Statement 4: Implement a Deterministic Finite Automaton (DFA) in “C‟ programming language that identifies the strings whose first alphabet is either 0 or 1. From second alphabet onwards the input string can have any number of input alphabets {either 0 or 1}.**

**Input from keyboard:** DFA parameters, and Input String

**Output on the screen:** If the input string is starting from 0, then the output should be “The String is starting from the input alphabet 0” If the input string is starting from 1, then the output should be “The String is starting from the input alphabet 1”

**Description of the problem statement:** Here we are creating a DFA to accept all strings which starts with either 0 or 1.

DFA and Transition table.

**Algorithm:**

**START**

1. READ: States, Input\_Alphabet, final\_states, Intial\_state, Transition table

2. READ input\_string (string)

3. CALL logic FUNCTION

INTITAILZE dfa = -1

WHILE (\*string != '\0') DO

IF(dfa == -1) THEN

dfa = 0; END IF

FOR j = 0; j < size of Input Alphabet; j++ DO

IF (\*string == Input\_Alphabet [j]) THEN

dfa = Transition table[dfa][j]; END IF

END FOR

string++;

END WHILE

return dfa;

5. IF dfa==final\_state THEN

IF (string[0]==’1’) THEN

PRINT: “The String is starting from the input alphabet 1”;

ELSE

PRINT: “The String is starting from the input alphabet 0”;

**STOP**

**C code**

#include<stdio.h>

int logic( char \*str, char \*inAlpha, int transTable[2][2]);

void main()

{

printf("Enter total states: ");

int Q[2];

for( int i=0; i<2; i++)

{

scanf("%d",&Q[i]);

}

int initialState;

printf("\nEnter Initial state: ");

scanf("%d",&initialState);

int finalState;

printf("\nEnter final state: ");

scanf("%d",&finalState);

char inAlpha[2+1];

printf("\nEnter the input alphabets: ");

scanf("%s",inAlpha);

char string[100];

printf("\nenter the string that is to be checked: ");

scanf("%s",string);

int transTable[2][2]={{1,1},{1,1}};

int dfa = logic(string, inAlpha, transTable);

if(dfa==finalState)

{

if(string[0]=='1'){

printf("\nThe String is starting from input Alphabet 1");

}

else

{

printf("\nThe String is starting from input Alphabet 0");

}

}

}

int logic(char \*str, char \*inAlpha, int transTable[2][2]){

int dfa= -1;

int len= 2;

while(\*str!='\0')

{

if(dfa==-1)

{

dfa=0;

}

for(int j=0; j<=len; j++){

if(\*str==inAlpha[j])

{

dfa=transTable[dfa][j];

}

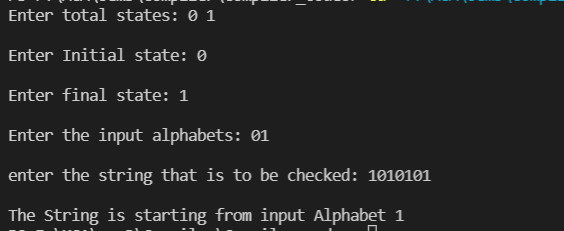
}

str++;

}

return dfa;

}



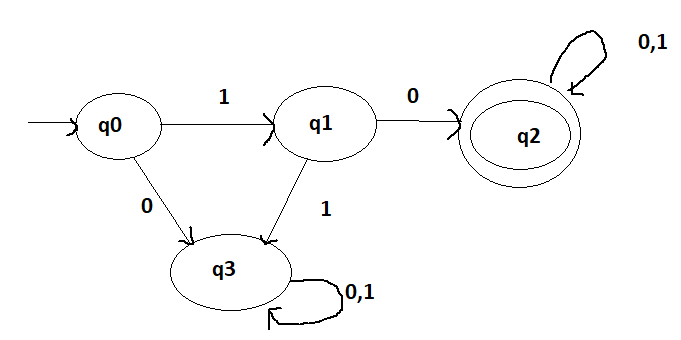
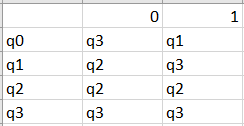
**Problem Statement 5: Implement a Deterministic Finite Automaton (DFA) in „C‟ programming language that identifies the strings whose first alphabet is 1 but the second alphabet is 0. From third alphabet onwards the input string can have any number of input alphabets {either 0 or 1}.**

**Input from keyboard:** DFA parameters, and Input String

**Output on the screen:** If the input string is as per the language given, then the output should be “The String belongs to the language L” If the input string is not as per the language given, then the output should be “The String does not belongs to the language L”.

**Description of the problem statement:** Here we are creating a DFA to accept all strings which starts with 10.

DFA and Transition Table

**Algorithm:**

**START**

1. READ: States, Input\_Alphabet, final\_states, Intial\_state, Transition table

2. READ input\_string (string)

3. CALL logic FUNCTION

INTITAILZE dfa = -1

WHILE (\*string != '\0') DO

IF(dfa == -1) THEN

dfa = 0; END IF

FOR j = 0; j < size of Input Alphabet; j++ DO

IF (\*string == Input\_Alphabet [j]) THEN

dfa = Transition table[dfa][j]; END IF

END FOR

string++;

END WHILE

return dfa;

5. IF dfa==final\_state THEN

PRINT: “The String belongs to the language L”;

ELSE

PRINT: “The String doesn’t belongs to the language L”;

**STOP**

C code:

#include<stdio.h>

int logic( char \*str, char \*inAlpha, int transTable[4][2]);

void main()

{

printf("Enter total states: ");

int Q[4];

for( int i=0; i<4; i++)

{

scanf("%d",&Q[i]);

}

int initialState;

printf("\nEnter Initial state: ");

scanf("%d",&initialState);

int finalState;

printf("\nEnter final state: ");

scanf("%d",&finalState);

char inAlpha[2+1];

printf("\nEnter the input alphabets: ");

scanf("%s",inAlpha);

char string[100];

printf("\nenter the string that is to be checked: ");

scanf("%s",string);

int transTable[4][2]={{3,1},{2,3},{2,2},{3,3}};

int dfa = logic(string, inAlpha, transTable);

if(dfa==finalState)

{

printf("\nThe String belongs to Language L");

}

else{

printf("\nThe String does not belong to Language L");

}

}

int logic(char \*str, char \*inAlpha, int transTable[4][2]){

int dfa= -1;

int len= 2;

while(\*str!='\0')

{

if(dfa==-1)

{

dfa=0;

}

for(int j=0; j<=len; j++){

if(\*str==inAlpha[j])

{

dfa=transTable[dfa][j];

}

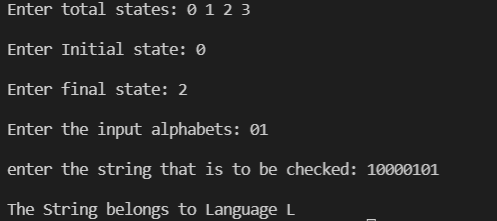
}

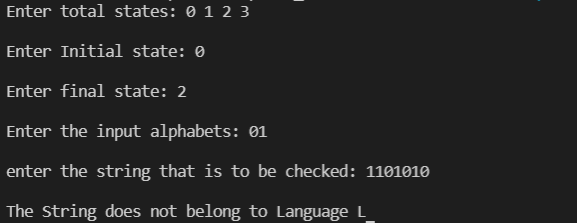
str++;

}

return dfa;

}





**Problem Statement 6: Implement a Deterministic Finite Automaton (DFA) in „C‟ programming language that identifies the strings whose first alphabet and the third alphabet is 1, but the second alphabet is either 0 or 1. From fourth alphabet onwards the input string can have any number of input alphabets {either 0 or 1}.**

**Input from keyboard:** DFA parameters, and Input String

**Output on the screen:** If the input string is as per the language given, then the output should be “The String is accepted” If the input string is as per the language given, then the output should be “The String is not accepted” .

**Description of the problem statement:** Here we are creating a DFA to accept all strings which starts with 101 or 111.

**Algorithm:**

**START**

1. READ: States, Input\_Alphabet, final\_states, Intial\_state, Transition table

2. READ input\_string (string)

3. CALL logic FUNCTION

INTITAILZE dfa = -1

WHILE (\*string != '\0') DO

IF(dfa == -1) THEN

dfa = 0; END IF

FOR j = 0; j < size of Input Alphabet; j++ DO

IF (\*string == Input\_Alphabet [j]) THEN

dfa = Transition table[dfa][j]; END IF

end for

string++;

END WHILE

return dfa;

5. IF dfa==final\_state

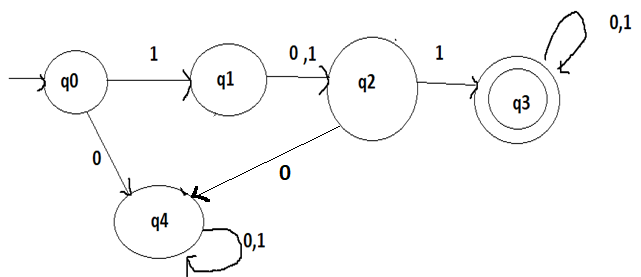
PRINT: “String is ACCEPTED”;

ELSE

PRINT: “String is Not Accepted”;

**STOP**

DFA and Transition Table

 Scatter chart

Description automatically generated

C code:

#include<stdio.h>

int logic( char \*str, char \*inAlpha, int transTable[5][2]);

void main()

{

printf("Enter total states: ");

int Q[5];

for( int i=0; i<5; i++)

{

scanf("%d",&Q[i]);

}

int initialState;

printf("\nEnter Initial state: ");

scanf("%d",&initialState);

int finalState;

printf("\nEnter final state: ");

scanf("%d",&finalState);

char inAlpha[2+1];

printf("\nEnter the input alphabets: ");

scanf("%s",inAlpha);

char string[100];

printf("\nenter the string that is to be checked: ");

scanf("%s",string);

int transTable[5][2]={{4,1},{2,2},{4,3},{3,3},{4,4}};

int dfa = logic(string, inAlpha, transTable);

if(dfa==finalState)

{

printf("\nThe String is accepted");

}

else{

printf("\nThe String is rejected");

}

}

int logic(char \*str, char \*inAlpha, int transTable[5][2]){

int dfa= -1;

int len= 2;

while(\*str!='\0')

{

if(dfa==-1)

{

dfa=0;

}

for(int j=0; j<=len; j++){

if(\*str==inAlpha[j])

{

dfa=transTable[dfa][j];

}

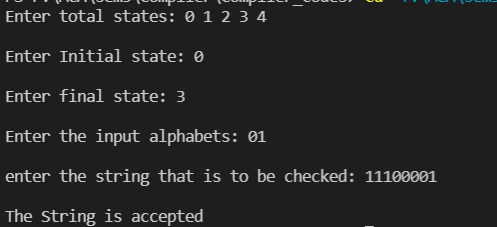
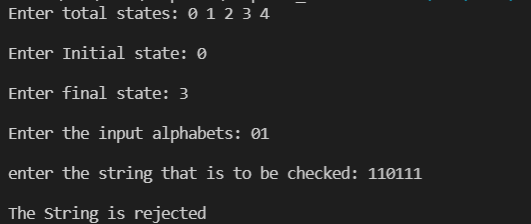
}

str++;

}

return dfa;

}

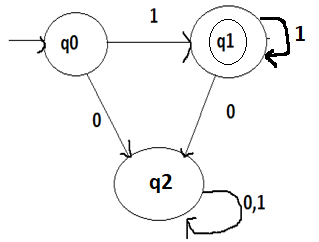
**Problem Statement 7: Implement a Deterministic Finite Automaton (DFA) in “C‟ programming language that identifies the strings which contains one or more numbers of input alphabets 1 only.**

**Input from keyboard:** DFA parameters, and Input String

**Output on the screen:** If all alphabets of the input string are 1 only, then the output should be “The String belongs to the language L” . If input string contains one or more number of 0, then the output should be “The String does not belongs to the language L”.

**Description of the problem statement:** Here we are creating a DFA to accept all strings which contains only 1. (eg: 111,1,11)

DFA and Transition Table

 Table

Description automatically generated

**Algorithm:**

**START**

1. READ: States, Input\_Alphabet, final\_states, Intial\_state, Transition table

2. READ input\_string (string)

3. CALL logic FUNCTION

INTITAILZE dfa = -1

WHILE (\*string != '\0') DO

IF(dfa == -1) THEN

dfa = 0; END IF

FOR j = 0; j < size of Input Alphabet; j++ DO

IF (\*string == Input\_Alphabet [j]) THEN

dfa = Transition table[dfa][j]; END IF

END FOR

string++;

END WHILE

return dfa;

5. IF dfa==final\_state

PRINT: “The String belongs to the language L”;

ELSE

PRINT: “The String doesn’t belongs to the language L”;

**STOP**

C code:

#include<stdio.h>

int logic( char \*str, char \*inAlpha, int transTable[3][2]);

void main()

{

printf("Enter total states: ");

int Q[3];

for( int i=0; i<3; i++)

{

scanf("%d",&Q[i]);

}

int initialState;

printf("\nEnter Initial state: ");

scanf("%d",&initialState);

int finalState;

printf("\nEnter final state: ");

scanf("%d",&finalState);

char inAlpha[2+1];

printf("\nEnter the input alphabets: ");

scanf("%s",inAlpha);

char string[100];

printf("\nenter the string that is to be checked: ");

scanf("%s",string);

int transTable[3][2]={{2,1},{2,1},{2,2}};

int dfa = logic(string, inAlpha, transTable);

if(dfa==finalState)

{

printf("\nThe String belongs to Language L");

}

else{

printf("\nThe String does not belong to Language L");

}

}

int logic(char \*str, char \*inAlpha, int transTable[3][2]){

int dfa= -1;

int len= 2;

while(\*str!='\0')

{

if(dfa==-1)

{

dfa=0;

}

for(int j=0; j<=len; j++){

if(\*str==inAlpha[j])

{

dfa=transTable[dfa][j];

}

}

str++;

}

return dfa;

}

