**AIM :**

To write a C program to simulate a Non-Deterministic Finite Automata.

**ALGORTIHM :**

1. Get the following as input from the user.

i. Number of states in the NFA

ii. Number of symbols in the input alphabet and the symbols

iii. Number of final states and their names

2. Declare a 3-dimensional matrix to store the transitions and initialize

all the entries with -1

3. Get the transitions from every state for every input symbol from the

user and store it in the matrix.

For example, consider the NFA shown below.

There are 4 states 0, 1, 2 and 3

There are two input symbols a and b. As the array index always

starts with 0, we assume 0th symbol is a and 1st symbol is b.

The transitions will be stored in the matrix as follows:

From state 0, for input a, there are two transitions to state 0 and 1,

which can be stored in the matrix as

m[0][0][0]=0

m[0][0][1]=1

Similarly, the other transitions can be stored as follows:

m[0][1][0]=0 (From state 0, for input b, one transition is to state 0)

m[0][1][1]=2 (From state 0, for input b, next transition is to state 2)

m[1][1][0]=3 (From state 1, for input b, move to state 3)

m[2][0][0]=3 (From state 2, for input a, move to state 3)

m[3][0][0]=3 (From state 3, for input a, move to state 3)

m[3][1][0]=3 (From state 3, for input b, move to state 3)

All the other entries in the matrix will be -1 indicating no moves

4. Get the input string from the user.

5. Find the length of the input string.

6. Read the input string character by character.

7. Repeat step 8 for every character

8. Refer the transition table for the entry corresponding to the present

state and the current input symbol and update the next state. As

there can be more than one transition, the next state will be an array.

9. From every state in the next state array, find the list of new

transitions and update the next state array.

10. When we reach the end of the input, if at least one of the final states

is present in the next state array, it means there is a path to a final

state. So the input is accepted. Otherwise the input is not accepted.

PROGRAM:

#include<stdio.h>

#include<string.h>

int main()

{

int i,j,k,l,m,next\_state[20],n,mat[10][10][10],flag,p;

int num\_states,final\_state[5],num\_symbols,num\_final;

int present\_state[20],prev\_trans,new\_trans,exit(0);

char ch,input[20];

int symbol[5],inp,inp1;

printf("How many states in the NFA : ");

scanf("%d",&num\_states);

printf("How many symbols in the input alphabet : ");

scanf("%d",&num\_symbols);

for(i=0;i<num\_symbols;i++)

{

printf("Enter the input symbol %d : ",i+1);

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

}

printf("How many final states : ");

scanf("%d",&num\_final);

for(i=0;i<num\_final;i++)

{

printf("Enter the final state %d : ",i+1);

scanf("%d",&final\_state[i]);

}

//Initialize all entries with -1 in Transition table

for(i=0;i<10;i++)

{

for(j=0;j<10;j++)

{

for(k=0;k<10;k++)

{

mat[i][j][k]=-1;

}

}

}

//Get input from the user and fill the 3D transition table

for(i=0;i<num\_states;i++)

{

for(j=0;j<num\_symbols;j++)

{printf("How many transitions from state %d for the input %d :",i,symbol[j]);

scanf("%d",&n);

for(k=0;k<n;k++)

{

printf("Enter the transition %d from state %d for the input%d : ",k+1,i,symbol[j]);

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

}

}

}

printf("The transitions are stored as shown below\n");

for(i=0;i<10;i++)

{

for(j=0;j<10;j++)

{

for(k=0;k<10;k++)

{

if(mat[i][j][k]!=-1)

printf("mat[%d][%d][%d] = %d\n",i,j,k,mat[i][j][k]);

}

}

}

while(1)

{

printf("Enter the input string : ");

scanf("%s",input);

present\_state[0]=0;

prev\_trans=1;

l=strlen(input);

for(i=0;i<l;i++)

{

if(input[i]=='0')

inp1=0;

else if(input[i]=='1')

inp1=1;

else

{

printf("Invalid input\n");

}

for(m=0;m<num\_symbols;m++)

{

if(inp1==symbol[m])

{

inp=m;

break;

}

}

new\_trans=0;

for(j=0;j<prev\_trans;j++)

{

k=0;

p=present\_state[j];

while(mat[p][inp][k]!=-1)

{

next\_state[new\_trans++]=mat[p][inp][k];

k++;

}

}

for(j=0;j<new\_trans;j++)

{

present\_state[j]=next\_state[j];

}

prev\_trans=new\_trans;

}

flag=0;

for(i=0;i<prev\_trans;i++)

{

for(j=0;j<num\_final;j++)

{

if(present\_state[i]==final\_state[j])

{

flag=1;

break;

}

}

}

if(flag==1)

printf("Acepted\n");

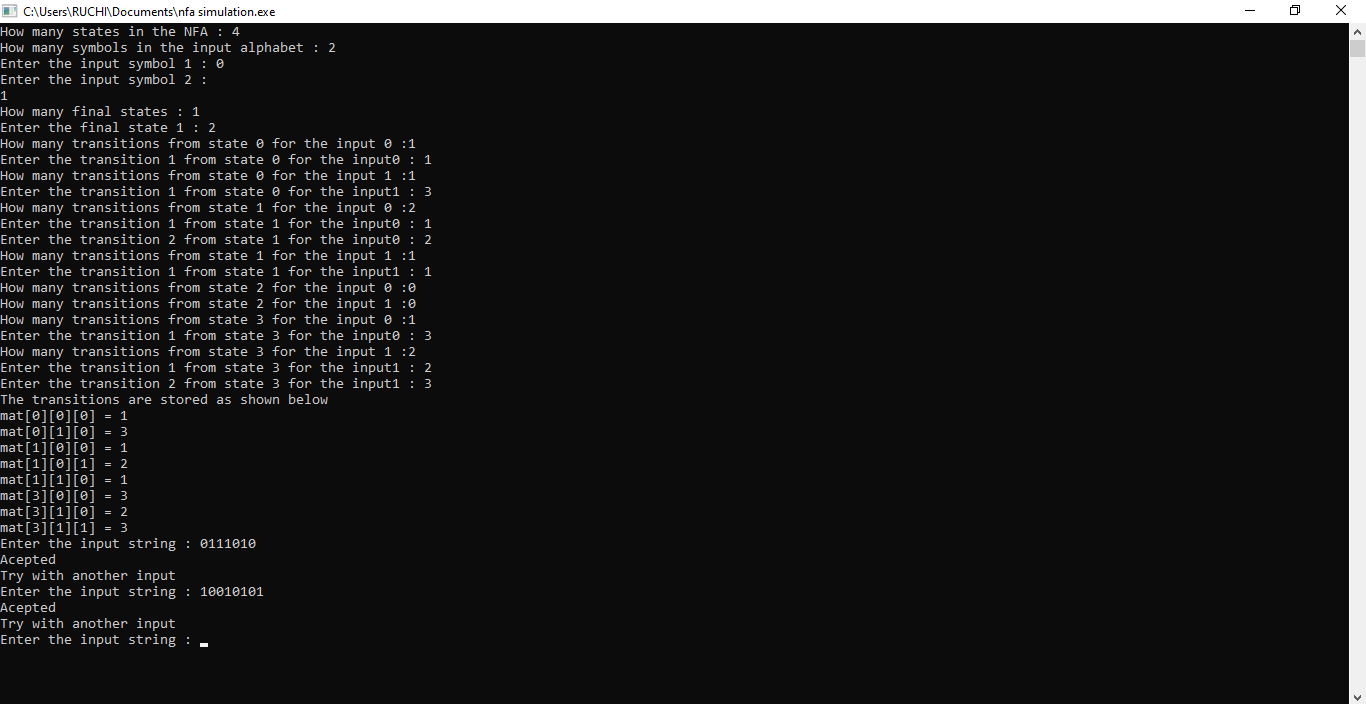
else

printf("Not accepted\n");

printf("Try with another input\n");

}

OUTPUT:



RESULT:

Thus executed successfully