1. Write a C program to simulate a Deterministic Finite Automata (DFA) for the given language representing strings that start with a and end with a

#include<stdio.h>

#include<string.h>

#define max 20

int main()

{

int trans\_table[4][2]={{1,3},{1,2},{1,2},{3,3}};

int final\_state=2,i;

int present\_state=0;

int next\_state=0;

int invalid=0;

char input\_string[max];

printf("Enter a string:");

scanf("%s",input\_string);

int l=strlen(input\_string);

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

{

if(input\_string[i]=='a')

next\_state=trans\_table[present\_state][0];

else if(input\_string[i]=='b')

next\_state=trans\_table[present\_state][1];

else

invalid=l;

present\_state=next\_state;

}

if(invalid==l)

{

printf("Invalid input");

}

else if(present\_state==final\_state)

printf("Accept\n");

else

printf("Don't Accept\n");

}

Input-abaab-accept

1. Write a C program to simulate a Deterministic Finite Automata (DFA) for the given language representing strings that start with 0 and end with 1

#include<stdio.h>

#define max 100

main() {

char str[max],f='a';

int i;

printf("enter the string to be checked: ");

scanf("%s",str);

for(i=0;str[i]!='\0';i++) {

switch(f) {

case 'a': if(str[i]=='0') f='b';

else if(str[i]=='1') f='a';

break;

case 'b': if(str[i]=='0') f='b';

else if(str[i]=='1') f='c';

break;

case 'c': if(str[i]=='0') f='b';

else if(str[i]=='1') f='a';

break;

}

}

if(f=='c')

printf("String is accepted", f);

else printf("String is not accepted", f);

return 0;

}

Input-00101

1. Write a C program to check whether a given string belongs to the language defined by a Context Free Grammar (CFG)

S → 0A1 A → 0A | 1A | ε

#include<stdio.h>

#include<string.h>

int main(){

char s[100];

int i,flag;

int l;

printf("enter a string to check:");

scanf("%s",s);

l=strlen(s);

flag=1;

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

{

if(s[i]!='0' && s[i]!='1')

{

flag=0;

}

}

if(flag!=1)

printf("string is Not Valid\n");

if(flag==1)

{

if (s[0]=='0'&&s[l-1]=='1')

printf("string is accepted\n");

else

printf("string is Not accepted\n");

}

}

Input-01

1. Write a C program to check whether a given string belongs to the language defined by a Context Free Grammar (CFG)

S → 0S0 | 1S1 | 0 | 1 | ε

#include<stdio.h>

#include<string.h>

int main()

{

char s[100];

int i,flag,flag1,a,b;

int l;

printf("enter a string to check:");

scanf("%s",s);

l=strlen(s);

flag=1;

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

{

if(s[i]!='0' && s[i]!='1')

{

flag=0;

}

}

if(flag!=1)

printf("string is Not Valid\n");

if(flag==1)

{

flag1=1;

a=0;b=l-1;

while(a!=(l/2))

{

if(s[a]!=s[b])

{

flag1=0;

}

a=a+1;

b=b-1;

}

if (flag1==1)

{

printf("The string is a palindrome\n");

printf("string is accepted\n");

}

else

{

printf("The string is not a palindrome\n");

printf("string is Not accepted\n");

}

}

}

Input-110101011

1. Write a C program to check whether a given string belongs to the language defined by a Context Free Grammar (CFG)

S → 0S0 | A A → 1A | ε

#include<stdio.h>

#include<string.h>

int main()

{

char s[100];

int i,flag,flag1,a,b;

int l,count1,count2;

printf("enter a string to check:");

scanf("%s",s);

l=strlen(s);

flag=1;

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

{

if(s[i]!='0' && s[i]!='1')

{

flag=0;

}

}

if(flag!=1)

printf("string is Not Valid\n");

if(flag==1)

{

i=0;count1=0;

while(s[i]=='0') // Count the no of 0s in the front

{

count1++;

i++;

}

while(s[i]=='1')

{

i++; // Skip all 1s

}

flag1=1;

count2=0;

while(i<l)

{

if(s[i]=='0')// Count the no of 0s at the end

{

count2++;

}

else

{

flag1=0;

}

i++;

}

if(flag1==1)

{

if(count1==count2)

{

printf("The string satisfies the condition 0n1m0n\n");

printf("String Accepted\n");

}

else

{

printf("The string does not satisfy the condition 0n1m0n\n");

printf("String Not Accepted\n");

}

}

else

{

printf("The string does not satisfy the condition 0n1m0n\n");

printf("String Not Accepted\n");

}

}

}

Input-00011000

1. Write a C program to check whether a given string belongs to the language defined by a Context Free Grammar (CFG)

S → 0S1 | ε

#include<stdio.h>

#include<string.h>

int main()

{

char s[100];

int i,flag,flag1,flag2;

int l;

printf("enter a string to check:");

scanf("%s",s);

l=strlen(s);

flag=1;

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

{

if(s[i]!='0' && s[i]!='1')

{

flag=0;

}

}

if(flag!=1)

printf("string is Not Valid\n");

if(flag==1)

{

if(l%2!=0) // If string length is odd

{

printf("The string does not satisfy the condition 0n1n\n");

printf("String Not Accepted\n");

}

else

{

// To check first half contains 0s

flag1=1;

for(i=0;i<(l/2);i++)

{

if(s[i]!='0')

{

flag1=0;

}

}

// To check second half contains 1s

flag2=1;

for(i=l/2;i<l;i++)

{

if(s[i]!='1')

{

flag2=0;

}

}

if(flag1==1 && flag2==1)

{

printf("The string satisfies the condition 0n1n\n");

printf("String Accepted\n");

}

else

{

printf("The string does not satisfy the condition 0n1n\n");

printf("String Not Accepted\n");

}

}

}

}

Input-0000011111

1. Write a C program to check whether a given string belongs to the language defined by a Context Free Grammar (CFG)

S → A101A, A → 0A | 1A | ε

#include<stdio.h>

#include<string.h>

int main()

{

char s[100];

int i,flag,flag1;

int l;

printf("enter a string to check:");

scanf("%s",s);

l=strlen(s);

flag=1;

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

{

if(s[i]!='0' && s[i]!='1')

{

flag=0;

}

}

if(flag==1)

printf("string is Valid\n");

else

printf("string is Not Valid\n");

if(flag==1)

{

flag1=0;

for(i=0;i<l-2;i++)

{

if(s[i]=='1')

{

if(s[i+1]=='0' && s[i+2]=='1')

{

flag1=1;

printf("Substring 101 exists. String accepted\n");

break;

}

}

}

if(flag1==0)

printf("Substring 101 does not exist. String not accepted\n");

}

}

Input-000111010100

1. Write a C program to simulate a Non-Deterministic Finite Automata (NFA) for the given language representing strings that start with b and end with a

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define MAX\_STRING\_LEN 100

// Function to simulate the NFA

int simulateNFA(char\* input) {

int state = 0;

int len = strlen(input);

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

char c = input[i];

if (state == 0) {

if (c == 'b') {

state = 1;

} else {

return 0;

}

} else {

if (c == 'a' && i == len - 1) {

return 1;

}

}

}

return 0;

}

int main() {

char input[MAX\_STRING\_LEN];

printf("Enter a string: ");

scanf("%s", input);

if (simulateNFA(input)) {

printf("Accepted\n");

} else {

printf("Rejected\n");

}

return 0;

}

Input-bbaa

1. Write a C program to simulate a Non-Deterministic Finite Automata (NFA) for the given language representing strings that start with o and end with 1

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define MAX\_STRING\_LEN 100

int simulateNFA(char\* input) {

int state = 0;

int len = strlen(input);

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

{

char c = input[i];

if (state == 0) {

if (c == '0') {

state = 1;

} else {

return 0;

}

} else {

if (c == '1' && i == len - 1) {

return 1;

}

}

}

return 0;

}

int main() {

char input[MAX\_STRING\_LEN];

printf("Enter a string: ");

scanf("%s", input);

if (simulateNFA(input)) {

printf("Accepted\n");

} else {

printf("Rejected\n");

}

return 0;

}

Input-010101

1. Write a C program to find ε -closure for all the states in a Non-Deterministic Finite Automata (NFA) with ε -moves.

#include<stdio.h>

#include<string.h>

int trans\_table[10][5][3];

char symbol[5],a;

int e\_closure[10][10],ptr,state;

void find\_e\_closure(int x);

int main()

{

int i,j,k,n,num\_states,num\_symbols;

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

{

for(j=0;j<5;j++)

{

for(k=0;k<3;k++)

{

trans\_table[i][j][k]=-1;

}

}

}

printf("How may states in the NFA with e-moves:");

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

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

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

printf("Enter the symbols without space. Give 'e' first:");

scanf("%s",symbol);

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

{

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

{

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

scanf("%d",&n);

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

{

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

scanf("%d",&trans\_table[i][j][k]);

}

}

}

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

{

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

{

e\_closure[i][j]=-1;

}

}

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

e\_closure[i][0]=i;

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

{

if(trans\_table[i][0][0]==-1)

continue;

else

{

state=i;

ptr=1;

find\_e\_closure(i);

}

}

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

{

printf("e-closure(%d)= {",i);

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

{

if(e\_closure[i][j]!=-1)

{

printf("%d, ",e\_closure[i][j]);

}

}

printf("}\n");

}

}

void find\_e\_closure(int x)

{

int i,j,y[10],num\_trans;

i=0;

while(trans\_table[x][0][i]!=-1)

{

y[i]=trans\_table[x][0][i];

i=i+1;

}

num\_trans=i;

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

{

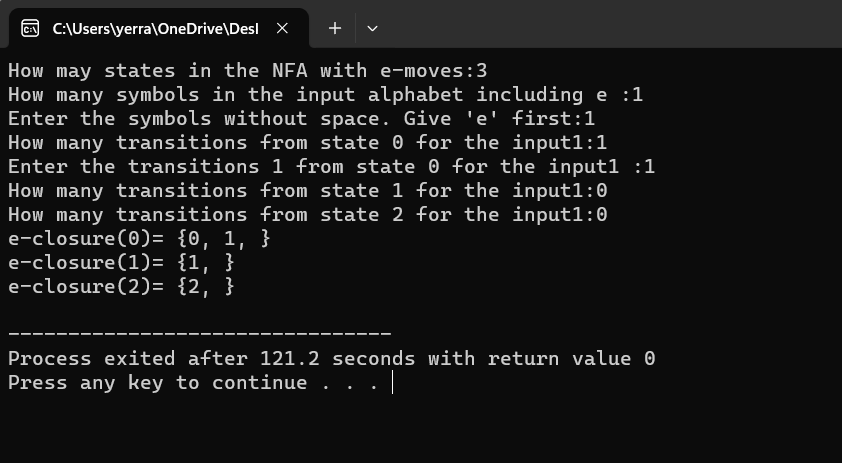
e\_closure[state][ptr]=y[j];

ptr++;

find\_e\_closure(y[j]);

}

}

Input-11. Write a C program to find ε -closure for all the states in a Non-Deterministic Finite Automata (NFA) with ε -moves.

#include <stdio.h>

#include <stdlib.h>

#define MAX\_STATES 50

#define MAX\_ALPHABET 50

void epsilonClosure(int state, int n, int epsilon[][MAX\_STATES], int closure[]) {

int i;

closure[state] = 1;

for (i = 0; i < n; i++) {

if (epsilon[state][i] == 1 && closure[i] == 0) {

epsilonClosure(i, n, epsilon, closure);

}

}

}

int main() {

int n, m; // n = number of states, m = alphabet size

printf("Enter the number of states: ");

scanf("%d", &n);

printf("Enter the alphabet size: ");

scanf("%d", &m);

int epsilon[MAX\_STATES][MAX\_STATES] = {0}; // ε-moves

int closure[MAX\_STATES] = {0};

printf("Enter the transition table:\n");

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

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

printf("Transition for state %d with input %d (use -1 for no transition): ", i, j);

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

}

}

printf("Enter the ε-moves (enter -1 to finish):\n");

int x, y;

while (1) {

printf("Enter ε-move (from state to state, -1 to finish): ");

scanf("%d %d", &x, &y);

if (x == -1 || y == -1) {

break;

}

epsilon[x][y] = 1;

}

printf("\nε-closure for each state:\n");

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

// Reset closure array for each state

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

closure[j] = 0;

}

epsilonClosure(i, n, epsilon, closure);

printf("ε-closure(%d): { ", i);

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

if (closure[j] == 1) {

printf("%d ", j);

}

}

printf("}\n");

}

return 0;

}