**CSA1375**

**THEORY OF COMPUTATION**

**(LAB PROGRAMS)**

**[C-Programming]**

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

AIM: To 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

ALGORITHM:

1.Start the code by including header files.

2.Declare an array str to hold the input.

3.Use the while loop to iterate through each character until final state is reached.

4.If the final state is reached print the message that final state is reached.

5.End the program

PROGRAM:

#include <stdio.h>

int main() {

char str[100];

int state = 0, i = 0;

printf("Enter a string: ");

scanf("%s", str);

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

if (state == 0 && str[i] == 'a') {

state = 1;

} else if (state == 1 && str[i] != 'a') {

state = 2;

} else if (state == 2) {

break;

}

i++;

}

if (state == 2) {

printf("String is not accepted by the DFA.\n");

} else {

printf("String is accepted by the DFA.\n");

}

return 0;

}

SAMPLE INPUT/OUTPUT:

ENTER A STRING: aba

String is accepted by dfa.

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

AIM: To 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

ALGORITHM:

1.Start the code by including header files.

2.Declare an array str to hold the input.

3.Use the while loop to iterate through each character until final state is

reached.

4.If the final state is reached print the message that final state is reached.

5.End the program

PROGRAM:

#include <stdio.h>

int main() {

char str[100];

int state = 0, i = 0;

printf("Enter a string: ");

scanf("%s", str);

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

if (state == 0 && str[i] == '0') {

state = 1;

} else if (state == 1 && str[i] != '1') {

state = 2;

} else if (state == 2) {

break;

}

i++;

}

if (state == 2) {

printf("String is not accepted by the DFA.\n");

} else {

printf("String is accepted by the DFA.\n");

}

return 0;

}

SAMPLE INPUT/OUTPUT:

ENTER A STRING: aba

String is accepted by dfa.

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

AIM: To 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

ALGORITHM:

1.Start the code by including header files.

2.Declare a function checkString() to check input validity.

3.Use if else condition to check each character until final state is reached.

4.If the final state is reached print the message that final state is reached.

5.End the program

PROGRAM:

#include <stdio.h>

#include <string.h>

int checkString(char str[]) {

int len = strlen(str);

if (str[0] == '0' && str[len-1] == '1') {

str[len-1] = '\0';

str++;

if (strlen(str) == 0) {

return 1;

} else {

return checkString(str);

}

} else {

return 0;

}

}

int main() {

char str[100];

printf("Enter a string: ");

scanf("%s", str);

if (checkString(str)) {

printf("The string belongs to the CFG language.\n");

} else {

printf("The string does not belong to the CFG language.\n");

}

return 0;

}

SAMPLE INPUT/OUTPUT:

ENTER A STRING: 0101

The string belongs to cfg language.

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

AIM: To write a c program to check whether a given string belongs to the above mentioned CFG

ALGORITHM:

1.Start the code by including header files.

2.Declare a function checkString() to check input validity.

3.Use if else condition to check each character until final state is reached.

4.If the final state is reached print the message that final state is reached.

5.End the program

PROGRAM:

#include <stdio.h>

#include <string.h>

int checkString(char str[]) {

int len = strlen(str);

if (len == 0 || (len == 1 && (str[0] == '0' || str[0] == '1'))) {

return 1; // The string belongs to the CFG language

}

Zif (str[0] == str[len-1]) {

str[len-1] = '\0';

str++;

if (checkString(str)) {

return 1;

}

}

if (len >= 3 && str[0] == '0' && str[len-1] == '0') {

// Remove the first and last characters from the string

str[len-1] = '\0';

str++;

if (checkString(str)) {

return 1; // The string belongs to the CFG language

}

}

if (len >= 3 && str[0] == '1' && str[len-1] == '1') {

str[len-1] = '\0';

str++;

if (checkString(str)) {

return 1;

}

}

return 0;

}

int main() {

char str[100];

printf("Enter a string: ");

scanf("%s", str);

if (checkString(str)) {

printf("The string belongs to the CFG language.\n");

} else {

printf("The string does not belong to the CFG language.\n");

}

return 0;

}

SAMPLE INPUT/OUTPUT:

ENTER A STRING: 010

The string belongs to cfg language.

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

AIM: To write a c program to check whether a given string belongs to the

given CFG language

ALGORITHM:

1.Start the code by including header files.

2.Declare a function checkString() to check input validity.

3.Use if else condition to check each character until final state is reached.

4.If the final state is reached print the message that final state is reached.

5.End the program

PROGRAM:

#include <stdio.h>

#include <string.h>

int checkString(char str[]) {

int len = strlen(str);

if (len == 0 || (len == 1 && str[0] == 'A')) {

return 1; // The string belongs to the CFG language

}

if (len >= 3 && str[0] == '0' && str[len-1] == '0') {

str[len-1] = '\0';

str++;

if (checkString(str)) {

return 1;

}

}

if (len >= 2 && str[0] == '1') {

str++;

if (checkString(str)) {

return 1; // The string belongs to the CFG language

}

}

return 0;

}

int main() {

char str[100];

printf("Enter a string: ");

scanf("%s", str);

if (checkString(str)) {

printf("The string belongs to the CFG language.\n");

} else {

printf("The string does not belong to the CFG language.\n");

}

return 0;

}

SAMPLE INPUT/OUTPUT:

ENTER A STRING: 010

The string belongs to cfg language

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

AIM: To write a c program to check whether a given string belongs to the

given CFG language.

ALGORITHM:

1.Start the code by including header files.

2.Declare a function checkString() to check input validity.

3.Use if else condition to check each character until final state is reached.

4.If the final state is reached print the message that final state is reached.

5.End the program

PROGRAM:

#include <stdio.h>

#include <string.h>

int checkString(char str[]) {

int len = strlen(str);

if (len == 0) {

return 1;

}

if (len >= 3 && str[0] == '0' && str[len-1] == '1') {

int index = len-1;

while (index >= 0 && str[index] != '1') {

index--;

}

if (index > 0 && checkString(str+1) && checkString(str+index+1)) {

return 1;

}

}

if (len == 1 && str[0] == 'ε') {

return 1;

}

return 0;

}

int main() {

char str[100];

printf("Enter a string: ");

scanf("%s", str);

if (checkString(str)) {

printf("The string belongs to the CFG language.\n");

} else {

printf("The string does not belong to the CFG language.\n");

}

return 0;

}

SAMPLE INPUT/OUTPUT:

ENTER A STRING: 0101

The string belongs to cfg language

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

AIM: To write a c program to check whether a given string belongs to the

given CFG language.

ALGORITHM:

1.Start the code by including header files.

2.Declare a function checkString() to check input validity.

3.Use if else condition to check each character until final state is reached.

4.If the final state is reached print the message that final state is reached.

5.End the program

PROGRAM:

#include <stdio.h>

#include <string.h>

int checkString(char str[]) {

int len = strlen(str);

if (len == 0 || len < 4) {

return 0;

}

if (str[0] == 'A' && str[1] != '1' && str[len-1] == 'A' && str[len-2] == '0' && str[len-3] == '1' && checkString(str+2) && checkString(str+3) && checkString(str+4) && checkString(str+len-4)) {

return 1;

}

int i;

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

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

return 0;

}

}

return 1;

}

int main() {

char str[100];

printf("Enter a string: ");

scanf("%s", str);

if (checkString(str)) {

printf("The string belongs to the CFG language.\n");

} else {

printf("The string does not belong to the CFG language.\n");

}

return 0;

}

SAMPLE INPUT/OUTPUT:

ENTER A STRING: A010101A

The string belongs to cfg language

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

AIM: To write a c program to check whether a given string belongs to the

given NFA language.

ALGORITHM:

1.Start the code by including header files.

2.Declare a function checkString() to check input validity.

3.Use if else condition to check each character until final state is reached.

4.If the final state is reached print the message that final state is reached.

5.End the program

PROGRAM:

#include <stdio.h>

#define NUM\_STATES 3

#define NUM\_SYMBOLS 2

int transition\_table[NUM\_STATES][NUM\_SYMBOLS][NUM\_STATES] = {

{{0, 1}, {0, 0}},

{{0, 1}, {2, 0}},

{{0, 0}, {0, 0}}

};

int simulate\_nfa(char\* input\_string) {

int current\_state = 0;

int input\_index = 0;

while (input\_string[input\_index] != '\0') {

char input\_symbol = input\_string[input\_index];

int symbol\_index = (input\_symbol == 'b') ? 0 : 1;

int next\_state = transition\_table[current\_state][symbol\_index][0];

if (next\_state == -1) {

return 0; // no valid transition

}

current\_state = next\_state;

input\_index++;

}

return (current\_state == 2); // 2 is the accepting state

}

int main() {

char input\_string[100];

printf("Enter input string: ");

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

if (simulate\_nfa(input\_string)) {

printf("String accepted\n");

} else {

printf("String rejected\n");

}

return 0;

}

SAMPLE INPUT/OUTPUT:

ENTER A STRING: bababa

String accepted

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

AIM: To write a c program to check whether a given string belongs to the

given NFA language.

ALGORITHM:

1.Start the code by including header files.

2.Declare a function checkString() to check input validity.

3.Use if else condition to check each character until final state is reached.

4.If the final state is reached print the message that final state is reached.

5.End the program

PROGRAM:

#include <stdio.h>

#include <string.h>

#define NUM\_STATES 4

int delta[NUM\_STATES][2][2] = {

{{0, 1}, {-1, -1}},

{{-1, -1}, {2, -1}},

{{-1, 3}, {-1, -1}},

{{-1, -1}, {-1, -1}}

};

int simulate\_nfa(char \*str) {

int state = 0, i;

for (i = 0; i < strlen(str); i++) {

if (str[i] == 'o') {

state = delta[state][0][0];

} else if (str[i] == '1') {

state = delta[state][1][0];

} else {

printf("Invalid character '%c' in string\n", str[i]);

return -1;

}

if (state == -1) {

return 0;

}

}

return (state == 2);

}

int main() {

char str[100];

printf("Enter a string: ");

scanf("%s", str);

if (simulate\_nfa(str)) {

printf("String accepted\n");

} else {

printf("String rejected\n");

}

return 0;

}

SAMPLE INPUT/OUTPUT:

ENTER A STRING: 001

String accepted

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

AIM: To write a c program to find epsilon closures of all states in a NFA

ALGORITHM:

1.Start the code by including header files.

2.Declare a function checkString() to check input validity.

3.Use if else condition to check each character until final state is reached.

4.If the final state is reached print the message that final state is reached.

5.End the program

PROGRAM:

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define MAX\_STATES 10

int transitions[MAX\_STATES][MAX\_STATES];

int n;

int eps\_closure(int state, int\* closure) {

int i, j, cnt = 0;

closure[cnt++] = state;

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

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

if (transitions[closure[i]][j] == -1) {

continue;

}

if (j == state && transitions[closure[i]][j] == 0) {

closure[cnt++] = j;

} else if (transitions[closure[i]][j] == 0) {

int k, found = 0;

for (k = 0; k < cnt; k++) {

if (closure[k] == j) {

found = 1;

break;

}

}

if (!found) {

closure[cnt++] = j;

}

}

}

}

return cnt;

}

int main() {

int i, j, cnt;

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

scanf("%d", &n);

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

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

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

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

}

}

printf("ε-closure of each state:\n");

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

int closure[MAX\_STATES];

cnt = eps\_closure(i, closure);

printf("ε-closure(%d) = {", i);

for (j = 0; j < cnt; j++) {

printf("%d", closure[j]);

if (j < cnt - 1) {

printf(", ");

}

}

printf("}\n");

}

return 0;

}

SAMPLE INPUT/OUTPUT:

Input: Number of states: 4

Number of transitions: 6

Enter transition details: 1 a 2

1 ε 3

2 b 3

3 c 4

4 d 1

2 ε 4

Output: ε-Closure of state 1: {1, 3}

ε-Closure of state 2: {2, 4}

ε-Closure of state 3: {3}

ε-Closure of state 4: {4}