

**TOC LAB**

DFA and its Application

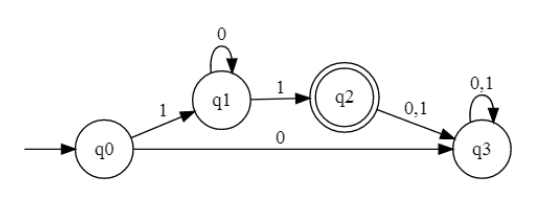
Submitted by Submitted to

Nabin Katwal Nischal Regmi

824 Department of Computer Science

Question 1

Write a program to simulate the DFA given in the figure. Simulate your DFA for different input strings and report the output.



#include <stdio.h>

#include <string.h>

int main() {

char \*str;

int i = 0;

int j;

int flag = 0;

int n;

printf("Enter String: ");

scanf("%s", str);

n = strlen(str);

while(1){

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

// state q1

i++;

if (i == n){

printf("String is not valid\n");

break;

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

// state q2

i++;

if (i == n){

printf("String is valid\n");

break;

}else {

// state q3

printf("String is invalid\n");

break;

}

}else if (str[i] == '0'){

// state q1

i++;

if (str[n-1] != '1') {

printf("String is invalid");

break;

}

j = i;

for(j; j< n-1;j++){

if (str[j] != '0'){

flag = 1;

}

}

if (flag == 1){

printf("String is invalid\n");

break;

}else{

printf("String is valid\n");

break;

}

}

} else {

printf("String is invalid\n");

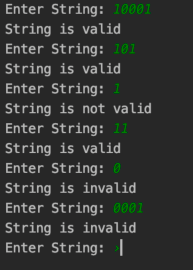
break;

}

}

return 0;

}



Question 2

An online portal requires its user to create passwords with following restrictions

(a) The password must be at least eight characters long

(b) The password must begin with a letter (a to z, A to z).

(c) The password must contain at least one special character and one number.

Write a program that checks whether a password complies with the above two rules.

#include <stdio.h>

#include <ctype.h>

int main(){

char pass[100];

int i = 0;

int currentState = 1;

printf("Enter your PASSWORD: ");

gets(pass);

while(pass[i]){

if (currentState == 1){

if(isdigit(pass[i]) || !isalpha(pass[i]))

{

currentState = 0;

break;

}else{

currentState = 2;

}

}else if (currentState >= 8 &&

currentState <= 13){

if(isdigit(pass[i]) || isalpha(pass[i]))

{

currentState == 13?

(currentState = 13) : (currentState+=1);

}else{

currentState += 12;

}

}else if(currentState >= 2 &&

currentState < 8){

if(isalpha(pass[i])){

currentState == 7?

(currentState=7) : (currentState += 1);

}else if(isdigit(pass[i])){

currentState += 6;

}else{

currentState += 12;

}

}else if(currentState >= 14 &&

currentState <= 19){

if (isdigit(pass[i])){

currentState += 6;

}else{

currentState == 19?

(currentState==19) : (currentState += 1);

}

}else if(currentState >= 20){

if(currentState==25){

break;

}else{

currentState += 1;

}

}

i++;

}

if(currentState == 25){

printf("The Password is

ACCEPTED!");

}else{

printf("The Password is not

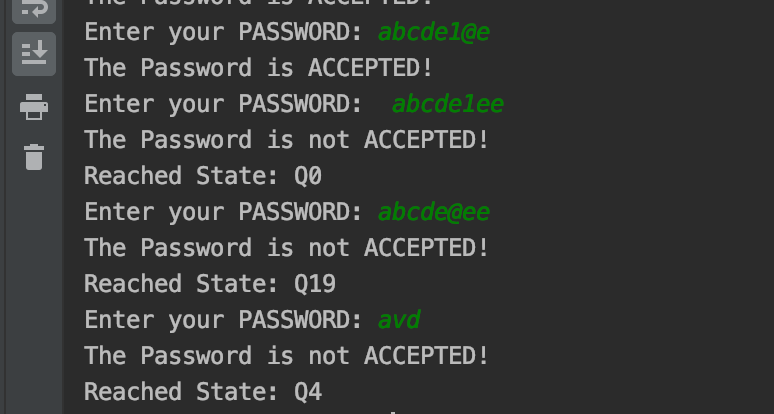
ACCEPTED!\n");

printf("Reached State: Q%d",

currentState);

}

}





**TOC LAB**

Simulate PDA

Submitted by Submitted to

Nabin Katwal Nischal Regmi

824 Department of Computer Science

Question

Write a C program to simulate the following Pushdown automata. You can also make the program more interactive by displaying how the machine moves from the initial configuration to an accepting or rejecting configuration.

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

int main() {

char\* str;

int n;

int i = 0;

int tos = 0;

char\* stack;

printf("Enter string: ");

scanf("%s", str);

n = strlen(str);

stack = (char\*)malloc(n\* sizeof(char));

if (\*(str+i) == '0'){

while(\*(str+i) == '0'){

\*(stack+tos++) = \*(str+i++);

}

while(\*(str+i) == '1'){

if(\*(str+i+1) == '0') break;

\*(stack+--tos) = 'b';

i++;

}

}

if (\*(stack+0) == 'b'){

printf("String valid");

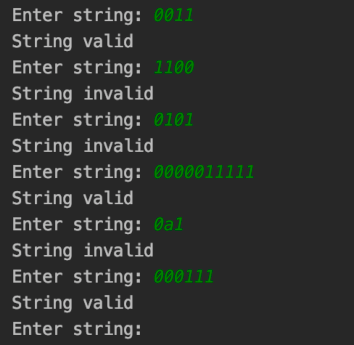
}else{

printf("String invalid");

}

return 0;

}



Question 2

Although it is possible in principle to simulate non-deterministic push- down automata, such a program would be computationally inefficient. For certain class of languages, we can construct deterministic PDA and simulate it. For the following grammar,

S →aSbS |bSaS |ε

Construct a PDA using the procedure we discussed in the lecture. Then write a program to simulate it. Note that the PDA would not still be deterministic, however, we can simulate it using ‘if-else’ wherever relevant.

#include <stdio.h>

#define N 32

void push(char stack[N], int \*top, char x) {

if(\*top < N - 1) {

(\*top)++;

stack[\*top] = x;

}

}

char pop(char stack[N], int \*top) {

char val;

if(\*top >= 0) {

val = stack[\*top];

stack[\*top] = '\0';

(\*top)--;

return val;

}

return 'E';

}

int nPDA(char stack[N], int \*top, char \*c)

{

printf("\nStack: %s, stackTop: %c, top:

%d, string: %s, char: %c", stack,

stack[\*top], \*top, c, \*c);

char stackCopy[N];

//pop the topmost element from stack

char stackTop = pop(stack, top);

//create a copy of stack and top

int topCopy = \*top;

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

stackCopy[i] = stack[i];

}

//flag to check if whether the input

string is accepted or rejected

int accepted = 0;

//base case of recursive function

if(\*c == '\0' && stackTop == 'E')

return 1;

else if(\*c == '\0' && stackTop != 'E')

return 0;

//check for (q, E, S/aSbS), (q, E, S/

bSaS) and (q, E, S/E)

if(\*c == 'E' && stackTop == 'S') {

push(stackCopy, &topCopy, 'S');

push(stackCopy, &topCopy, 'b');

push(stackCopy, &topCopy, 'S');

push(stackCopy, &topCopy, 'a');

accepted = nPDA(stackCopy, &topCopy, c+1);

if(accepted == 1) {

return 1;

} else {

//restore stackCopy and topCopy

topCopy = \*top;

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

stackCopy[i] = stack[i];

}

push(stackCopy, &topCopy, 'S');

push(stackCopy, &topCopy, 'a');

push(stackCopy, &topCopy, 'S');

push(stackCopy, &topCopy, 'b');

accepted = nPDA(stackCopy,

&topCopy, c+1);

if(accepted == 1) {

return 1;

} else {

//restore stackCopy and

topCopy

topCopy = \*top;

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

stackCopy[i] = stack[i];

}

return nPDA(stackCopy,

&topCopy, c+1);

}

}

} else if(\*c == 'a' && stackTop == 'a') {

return nPDA(stackCopy, &topCopy,

c+1);

} else if(\*c == 'b' && stackTop == 'b') {

return nPDA(stackCopy, &topCopy,

c+1);

}

return 0;

}

int main() {

char input[N], str[N\*2];

char \*c = str;

char stack[N];

int top = -1;

int accepted = 0;

push(stack, &top, 'S');

printf("Enter string: ");

scanf("%s", input);

// convert input string ab into EaEbE

for(int i = 0, j = 0; i < sizeof input /

sizeof input[0]; i++, j+= 2) {

str[j] = 'E';

str[j + 1] = input[i];

}

accepted = nPDA(stack, &top, c);

if(accepted)

printf("\naccepted");

else

printf("\nrejected");

return 0;

}

