**9. Write a program to implement PDA that accepts all strings over{1, 0} that have equal number of 0s and 1s.**

#include <stdio.h>

#include <string.h>

#define MAX 100

enum states

{

q0,

q1,

qf

};

void push(char ch);

void pop();

char get\_stack\_top();

enum states delta(enum states, char, char);

struct stack

{

char symbols[MAX];

int top;

};

struct stack s;

int main()

{

char input[20];

enum states curr\_state = q0;

s.top = -1;

int i = 0;

char ch = 'e'; // e indicating epsilon

char st\_top = 'e';

curr\_state = delta(curr\_state, ch, st\_top);

printf("\n Enter a binary string\t");

gets(input);

ch = input[i];

st\_top = get\_stack\_top();

int c = 0;

while (c <= strlen(input))

{

curr\_state = delta(curr\_state, ch, st\_top);

ch = input[++i];

st\_top = get\_stack\_top();

c++;

}

if (curr\_state == qf)

printf("\n The string %s is accepted.", input);

else

printf("\n The string %s is not accepted.", input);

return 0;

}

enum states delta(enum states s, char ch, char st\_top)

{

enum states curr\_state;

switch (s)

{

case q0:

if (ch == 'e' && st\_top == 'e')

{

curr\_state = q1;

push('$'); // $ is stack bottom marker

}

break;

case q1:

if (ch == '0' && (st\_top == '$' || st\_top == '0'))

{

curr\_state = q1;

push(ch);

}

else if (ch == '1' && (st\_top == '$' || st\_top == '1'))

{

curr\_state = q1;

push(ch);

}

else if (ch == '1' && st\_top == '0' || ch == '0' && st\_top == '1')

{

curr\_state = q1;

pop();

}

else if (ch == '\0' && st\_top == '$')

{

curr\_state = qf;

pop();

}

break;

}

return curr\_state;

}

// function to get stack top symbol

char get\_stack\_top()

{

return (s.symbols[s.top]);

}

// push function

void push(char ch)

{

if (s.top < MAX - 1)

{

s.symbols[++s.top] = ch;

}

else

{

printf("\n Stack Full.");

}

}

// pop function

void pop()

{

if (s.top > -1)

{

s.symbols[s.top] = ' ';

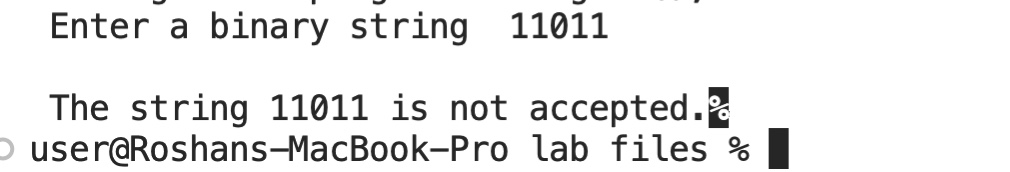
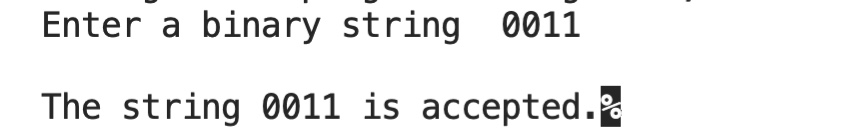
s.top--;

}

else

printf("\n Stack Empty.");

}

**OUTPUT**

**10. PDA accepting equal number of 0s and 1s with empty stack.**

#include <stdio.h>

#include <string.h>

#define MAX 100

enum states

{

q0

};

void push(char ch);

void pop();

char get\_stack\_top();

enum states delta(enum states, char, char);

struct stack

{

char symbols[MAX];

int top;

};

struct stack s;

int main()

{

char input[20];

enum states curr\_state = q0;

s.top = -1;

int i = 0;

char ch = 'e';

char st\_top = 'e';

curr\_state = delta(curr\_state, ch, st\_top);

printf("\n Enter a binary string\t");

gets(input);

ch = input[i];

st\_top = get\_stack\_top();

int c = 0;

while (c <= strlen(input))

{

curr\_state = delta(curr\_state, ch, st\_top);

ch = input[++i];

st\_top = get\_stack\_top();

c++;

}

if (s.symbols[s.top] == '$')

printf("\n The string %s is accepted.", input);

else

printf("\n The string %s is not accepted.", input);

return 0;

}

enum states delta(enum states s, char ch, char st\_top)

{

enum states curr\_state;

switch (s)

{

case q0:

if (ch == 'e' && st\_top == 'e')

{

curr\_state = q0;

push('$');

}

else if (ch == '0' && (st\_top == '$' || st\_top == '0'))

{

curr\_state = q0;

push(ch);

}

else if (ch == '1' && (st\_top == '$' || st\_top == '1'))

{

curr\_state = q0;

push(ch);

}

else if (ch == '1' && st\_top == '0' || ch == '0' && st\_top == '1')

{

curr\_state = q0;

pop();

}

else if (ch == '\0' && st\_top == '$')

{

curr\_state = q0;

// pop();

}

break;

}

return curr\_state;

}

char get\_stack\_top()

{

return (s.symbols[s.top]);

}

void push(char ch)

{

if (s.top < MAX - 1)

{

s.symbols[++s.top] = ch;

}

else

{

printf("\n Stack Full.");

}

}

void pop()

{

if (s.top > -1)

{

s.symbols[s.top] = ' ';

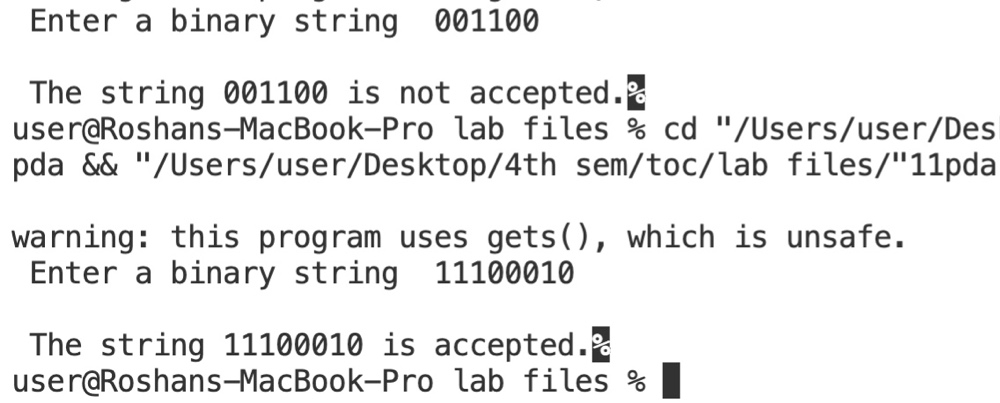
s.top--;

}

else

printf("\n Stack Empty.");

}

**OUTPUT**

**11. Implement the TM accepting the language{0n1 n / >= 1} over { 0, 1 }**

#include <stdio.h>

enum states

{

q0,

q1,

q2,

q3,

q4,

qr

};

int main()

{

char input[100];

enum states curr\_state = q0;

int i;

for (i = 0; i < 100; i++)

input[i] = '\0';

printf("\n Enter a binary string\t");

gets(input);

i = 0;

while (1)

{

switch (curr\_state)

{

case q0:

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

{

curr\_state = q1;

input[i] = 'x';

i++;

}

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

{

curr\_state = q3;

i++;

}

else

curr\_state = qr; // for invalid transition

break;

case q1:

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

{

curr\_state = q1;

i++;

}

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

{

curr\_state = q1;

i++;

}

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

{

curr\_state = q2;

input[i] = 'y';

i--;

}

else

curr\_state = qr;

break;

case q2:

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

{

curr\_state = q2;

i--;

}

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

{

curr\_state = q2;

i--;

}

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

{

curr\_state = q0;

i++;

}

else

curr\_state = qr;

break;

case q3:

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

{

curr\_state = q3;

i++;

}

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

{

curr\_state = q4;

}

else

curr\_state = qr;

break;

} // end of switch

if (curr\_state == qr || curr\_state == q4)

break;

} // end of while loop

if (curr\_state == q4)

printf("\n The string is accepted.");

else

printf("\n The string is not accepted.");

return 0;

}

**OUTPUT**

