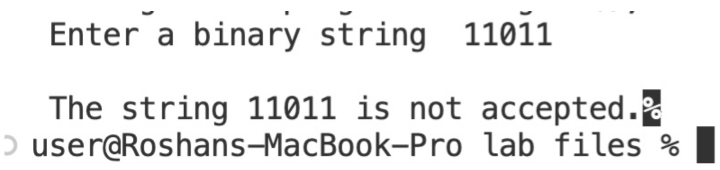
9. Write a program to implement PDA that accepts all strings over{1, 0} that have equal number of 0s and 1s.

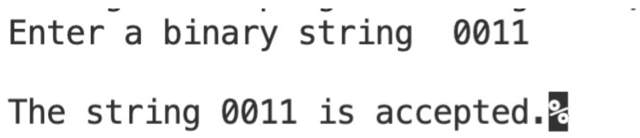
```
#include <stdio.h>
#include <string.h>
#define MAX 100
enum states
{
  q0,
  q1,
  qf
};
void push(char ch);
void pop();
char get_stack_top();
enum states delta(enum states, char, char);
struct stack
{
  char symbols[MAX];
  int top;
};
struct stack s;

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



```
Enter a binary string  11011

The string 11011 is not accepted.%
user@Roshans-MacBook-Pro lab files %
```



```
Enter a binary string  0011

The string 0011 is accepted.%
```

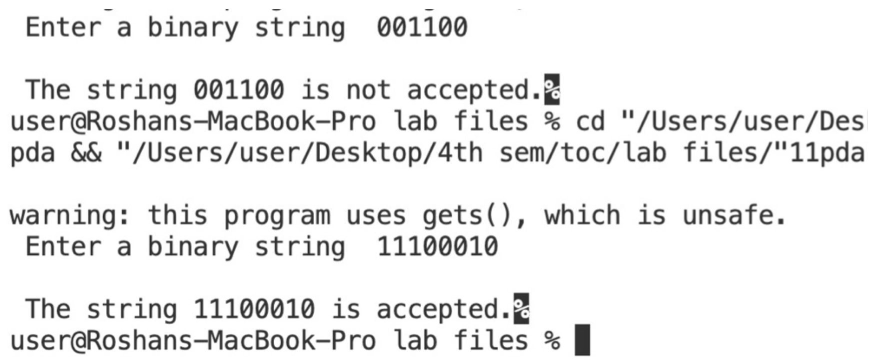
10. PDA accepting equal number of 0s and 1s with empty stack.

```
#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

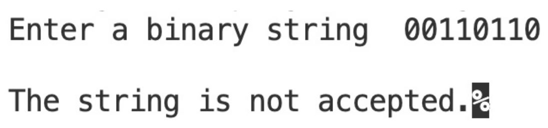


```
 Enter a binary string  001100

 The string 001100 is not accepted.%
user@Roshans-MacBook-Pro lab files % cd "/Users/user/Des
pda && "/Users/user/Desktop/4th sem/toc/lab files/"11pda

warning: this program uses gets(), which is unsafe.
 Enter a binary string  11100010

 The string 11100010 is accepted.%
user@Roshans-MacBook-Pro lab files % █
```

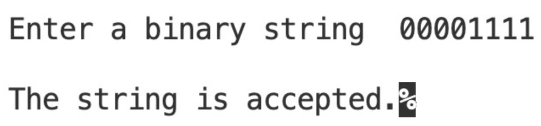
11. Implement the TM accepting the language{0n1 n / >= 1} over { 0, 1 }

```
#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



```
Enter a binary string  00110110

The string is not accepted.%
```



```
Enter a binary string  00001111

The string is accepted.%
```