**Write a program for generating regular expressions for regular grammar.**

import re

text = "horses are taller than dogs"

match = re.search (,text)

if match:

  print("Match found:", match.group())

  print("Match found:", match.group())

else:

  print("No match found.")

**Write a program for generating derivation sequence / language for the given sequence of productions**

def printarray(arr,size):

  for i in range(size):

    print (arr[i],end='')

    print()

    return

def getsuccessor(arr,k,n):

  p=k-1

  while (arr[p]==n and n<=p<k):

    p==1

    if (p<0):

      return 0

    arr[p]=arr[p]+1

    i=p+1

    while (i<k):

      i+=1

      return 1

def printsequence(n,k):

  arr=[0]\*k

  for i in range(k)

  arr[i]=1

  while true:

    printarray(arr,k)

    if getsuccessor(arr, k,n)==0:

      break

n=4

k=2

printsequence(n,k)

Design a Program for creating machine that accepts three consecutive one

# State A

def stateA(n):

  if(n[0]=='a'):

    stateB(n[1:])

  elif (n[0]=='b'):

    stateH(n[1:])

# State B

def stateB(n):

   if(len(n)== 0):

     print("String Not Accepted")

   else:

    if(n[0]=='a'):

      stateC(n[1:])

    elif (n[0]=='b'):

      stateI(n[1:])

# State C

def stateC(n):

  if(len(n)== 0):

    print("String Not Accepted")

  else:

    if(n[0]=='a'):

      stateD(n[1:])

    elif (n[0]=='b'):

      stateJ(n[1:])

# State D

def stateD(n):

  if(len(n)== 0):

    print("String Not Accepted")

  else:

    if(n[0]=='a'):

      stateQ2(n)

    elif (n[0]=='b'):

      stateE(n[1:])

# State E

def stateE(n):

   if(len(n)== 0):

     print("String Not Accepted")

   else:

     if(n[0]=='a'):

       stateQ2(n)

     elif (n[0]=='b'):

       stateF(n[1:])

# State F

def stateF(n):

  if(len(n)== 0):

    print("String Not Accepted")

  else:

    if(n[0]=='a'):

      stateQ2(n[1:])

    elif (n[0]=='b'):

      stateG(n[1:])

# State G

def stateG(n):

  if(len(n)== 0):

    print("String Accepted")

  else:

    if(n[0]=='a'):

      stateQ2(n)

    elif (n[0]=='b'):

      stateQ2(n)

# State H

def stateH(n):

  if(len(n)== 0):

    print("String Not Accepted")

  else:

    if(n[0]=='a'):

      stateI(n[1:])

    elif (n[0]=='b'):

      stateK(n[1:])

# State I

def stateI(n):

  if(len(n)== 0):

    print("String Not Accepted")

  else:

    if(n[0]=='a'):

      stateJ(n[1:])

    elif (n[0]=='b'):

      stateL(n[1:])

# State J

def stateJ(n):

  if(len(n)== 0):

    print("String Not Accepted")

  else:

    if(n[0]=='a'):

      stateE(n[1:])

    elif (n[0]=='b'):

      stateM(n[1:])

# State K

def stateK(n):

  if(len(n)== 0):

    print("String Not Accepted")

  else:

    if(n[0]=='a'):

      stateL(n[1:])

    elif (n[0]=='b'):

      stateN(n[1:])

# State L

def stateL(n):

  if(len(n)== 0):

    print("String Not Accepted")

  else:

    if(n[0]=='a'):

      stateM(n[1:])

    elif (n[0]=='b'):

      stateO(n[1:])

# State M

def stateM(n):

  if(len(n)== 0):

    print("String Not Accepted")

  else:

    if(n[0]=='a'):

      stateF(n[1:])

    elif (n[0]=='b'):

      stateP(n[1:])

# State N

def stateN(n):

  if(len(n)== 0):

    print("String Not Accepted")

  else:

    if(n[0]=='a'):

      stateO(n[1:])

    elif (n[0]=='b'):

      stateQ1(n)

# State O

def stateO(n):

  if(len(n)== 0):

    print("String Not Accepted")

  else:

    if(n[0]=='a'):

      stateP(n[1:])

    elif (n[0]=='b'):

      stateQ1(n)

# State P

def stateP(n):

  if(len(n)== 0):

    print("String Not Accepted")

  else:

    if(n[0]=='a'):

      stateG(n[1:])

    elif (n[0]=='b'):

      stateQ1(n[1:])

# State Q1

def stateQ1(n):

  print("String Not Accepted")

# State Q2

def stateQ2(n):

  print("String Not Accepted")

# take string input

n = "aaabb"

# call stateA

# to check the input

stateA(n)

**Design a program for accepting decimal number divisible by 2**

def stateq0(n):

#if length found 0

#print not accepted

if (len(n)==0):

print("string accepted")

else:

#if at index 0

#'0' found call

#function stateq0

if(n[0]=='0'):

stateq0(n[1:])

#else if '1' found

#call function q1.

elif (n[0]=='1'):

stateq1(n[1:])

def stateq1(n):

#if length found 0

#print not accepted

if (len(n)==0):

print("string not accepted")

else:

#if at index 0

#'0' found call

#function stateq0

if(n[0]=='0'):

stateq0(n[1:])

#else if '1' found

#call function q1.

elif (n[0]=='1'):

stateq1(n[1:])

#take number from user

n=int(input())

#converting number to binary

n = bin(n).replace("0b", "")

#call stateA

#to check the input

stateq0(n)

INPUT: 5

OUTPUT: String Not Accepted

* **Design a program for creating a machine which accepts string having equal no. of 1’s, 2’s and 0’s.**

# Python3 program to find subString with equal

# number of 0's, 1's and 2's

# Method to count number of subString which

# has equal 0, 1 and 2

**def** getSubStringWithEqual012(s) :

    arr **=** [];

    n **=** len(s);

    # generating subarrays

**for** i **in** range(n):

**for** j **in** range(i, n):

            s1 **=** ""

**for** k **in** range(i, 1 **+** j):

                s1**+=**s[k];

            arr.append(s1);

    count **=** 0;

    #    iterating over array of all subStrings

**for** i **in** range(len(arr)):

        countZero**=**0;

        countOnes**=**0;

        countTwo**=**0;

        curs **=** arr[i];

**for** j **in** range(len(curs)):

**if**(curs[j] **==** '0'):

                countZero**+=**1;

**if**(curs[j] **==** '1'):

                countOnes**+=**1;

**if**(curs[j] **==** '2'):

                countTwo**+=**1;

        # if number of ones,two and zero are equal in a subString

**if**(countZero **==** countOnes **and** countOnes **==** countTwo):

            count **+=** 1;

**return** count;

#  Driver's code

Str **=** "0102010";

# Function call

print(getSubStringWithEqual012(Str));

**Design a program for creating a machine which count number of 1’s and 0’s in a given string.**

**def** countSubstring(S, n) :

    # To store the total count

    # of substrings

    ans **=** 0;

    i **=** 0;

    # Traversing the string

**while** (i < n) :

        # Count of consecutive

        # 0's & 1's

        cnt0 **=** 0; cnt1 **=** 0;

        # Counting subarrays of

        # type "01"

**if** (S[i] **==** '0') :

            # Count the consecutive

            # 0's

**while** (i < n **and** S[i] **==** '0') :

                cnt0 **+=** 1;

                i **+=** 1;

            # If consecutive 0's

            # ends then check for

            # consecutive 1's

            j **=** i;

            # Counting consecutive 1's

**while** (j < n **and** S[j] **==** '1') :

                cnt1 **+=** 1;

                j **+=** 1;

        # Counting subarrays of

        # type "10"

**else** :

            # Count consecutive 1's

**while** (i < n **and** S[i] **==** '1') :

                cnt1 **+=** 1;

                i **+=** 1;

            # If consecutive 1's

            # ends then check for

            # consecutive 0's

            j **=** i;

            # Count consecutive 0's

**while** (j < n **and** S[j] **==** '0') :

                cnt0 **+=** 1;

                j **+=** 1;

        # Update the total count

        # of substrings with

        # minimum of (cnt0, cnt1)

        ans **+=** min(cnt0, cnt1);

    # Return answer

**return** ans;

# Driver code

**if** \_\_name\_\_ **==** "\_\_main\_\_" :

    S **=** "0001110010";

    n **=** len(S);

    # Function to print the

    # count of substrings

    print(countSubstring(S, n));

OR

cnt **=** 0

**def** count\_zero(s: str):

**global** cnt

**for** i **in** range(len(s)):

        div **=** int(s[i])

**if** div **==** 0:

            cnt **+=** 1

    print(cnt)

s **=** "101101"

count\_zero(s)

output:

2

Design a Program for creating machine that accepts the string always ending with 101.

# End position is checked using the string

# length value.

# q0 is the starting state.

# q1 and q2 are intermediate states.

# q3 and q4 are final states.

def q1(s, i) :

print("q1->", end="");

if (i == len(s)) :

print("NO");

return;

# state transitions

# 0 takes to q1, 1 takes to q3

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

q1(s, i + 1);

else :

q3(s, i + 1);

def q2(s, i) :

print("q2->", end = "");

if (i == len(s)) :

print("NO");

return;

# state transitions

# 0 takes to q4, 1 takes to q2

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

q4(s, i + 1);

else :

q2(s, i + 1);

def q3(s, i) :

print("q3->", end = "");

if (i == len(s)) :

print("YES");

return;

# state transitions

# 0 takes to q4, 1 takes to q2

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

q4(s, i + 1);

else :

q2(s, i + 1);

def q4(s, i) :

print("q4->", end = "");

if (i == len(s)) :

print("YES");

return;

# state transitions

# 0 takes to q1, 1 takes to q3

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

q1(s, i + 1);

else :

q3(s, i + 1);

def q0( s, i) :

print("q0->", end = "");

if (i == len(s)) :

print("NO");

return;

# state transitions

# 0 takes to q1, 1 takes to q2

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

q1(s, i + 1);

else :

q2(s, i + 1);

# Driver Code

if \_\_name\_\_ == "\_\_main\_\_" :

s = "010101";

# all state transitions are printed.

# if string is accpetable, YES is printed.

# else NO is printed

print("State transitions are", end = " ");

q0(s, 0);