Theory Of Computation – Practical Manual

SYBSC CS SEM IV:

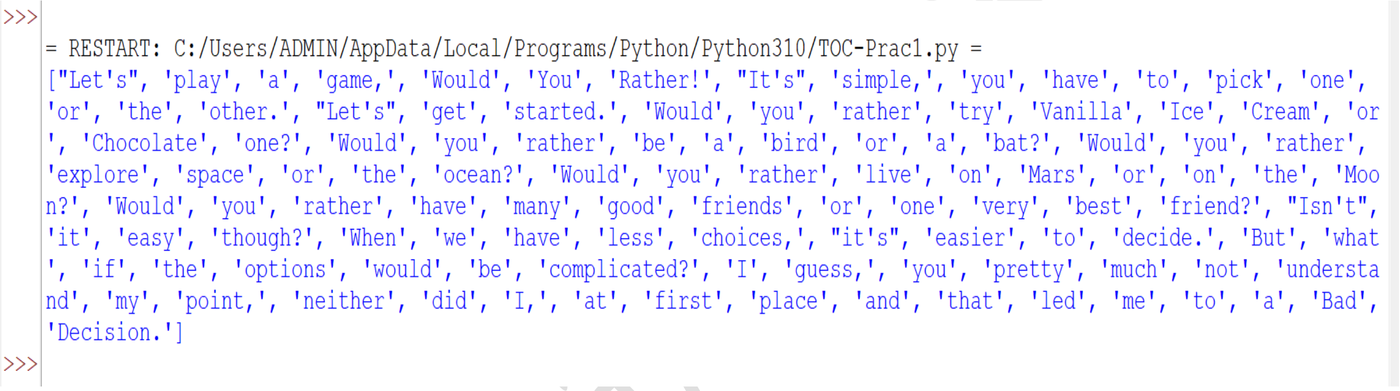
By: prof.Ajay Pashankar

Theory Of Computation – Practical Manual

1} Write a program for tokenization of given input in python.

```
my_text = """Let's play a game, Would You Rather! It's simple, you have to pick one or the other. Let's get started. Would you rather try Vanilla Ice Cream or Chocolate one? Would you rather be a bird or a bat? Would you rather explore space or the ocean? Would you rather live on Mars or on the Moon? Would you rather have many good friends or one very best friend? Isn't it easy though? When we have less choices, it's easier to decide. But what if the options would be complicated? I guess, you pretty much not understand my point, neither did I, at first place and that led me to a Bad Decision."""

print(my_text.split())
```

Output-

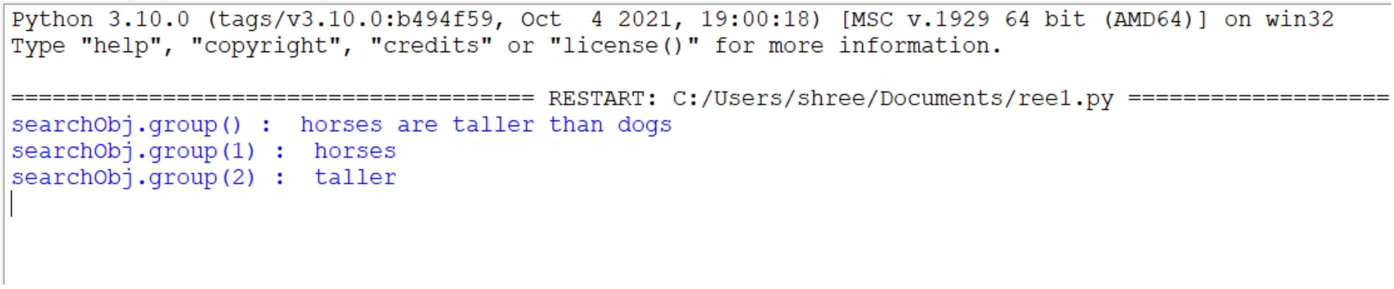


```
>>>
= RESTART: C:/Users/ADMIN/AppData/Local/Programs/Python/Python310/TOC-Prac1.py =
["Let's", 'play', 'a', 'game,', 'Would', 'You', 'Rather!', "It's", 'simple,', 'you', 'have', 'to', 'pick', 'one',
'or', 'the', 'other.', "Let's", 'get', 'started.', 'Would', 'you', 'rather', 'try', 'Vanilla', 'Ice', 'Cream', 'or
', 'Chocolate', 'one?', 'Would', 'you', 'rather', 'be', 'a', 'bird', 'or', 'a', 'bat?', 'Would', 'you', 'rather',
'explore', 'space', 'or', 'the', 'ocean?', 'Would', 'you', 'rather', 'live', 'on', 'Mars', 'or', 'on', 'the', 'Moo
n?', 'Would', 'you', 'rather', 'have', 'many', 'good', 'friends', 'or', 'one', 'very', 'best', 'friend?', "Isn't",
'it', 'easy', 'though?', 'When', 'we', 'have', 'less', 'choices,', "it's", 'easier', 'to', 'decide.', 'But', 'what
', 'if', 'the', 'options', 'would', 'be', 'complicated?', 'I', 'guess,', 'you', 'pretty', 'much', 'not', 'understa
nd', 'my', 'point,', 'neither', 'did', 'I,', 'at', 'first', 'place', 'and', 'that', 'led', 'me', 'to', 'a', 'Bad',
'Decision.']
>>>
```

2} Write a program for generating regular expressions for regular grammar in Python.

```
import re

line = "horses are taller than dogs";

searchObj = re.search( r'(.*) are (.*?) .*', line, re.M|re.I)

if searchObj:
   print ("searchObj.group() : ", searchObj.group())
   print ("searchObj.group(1) : ", searchObj.group(1))
   print ("searchObj.group(2) : ", searchObj.group(2))
else:
   print ("Nothing found!!")
```

Output-



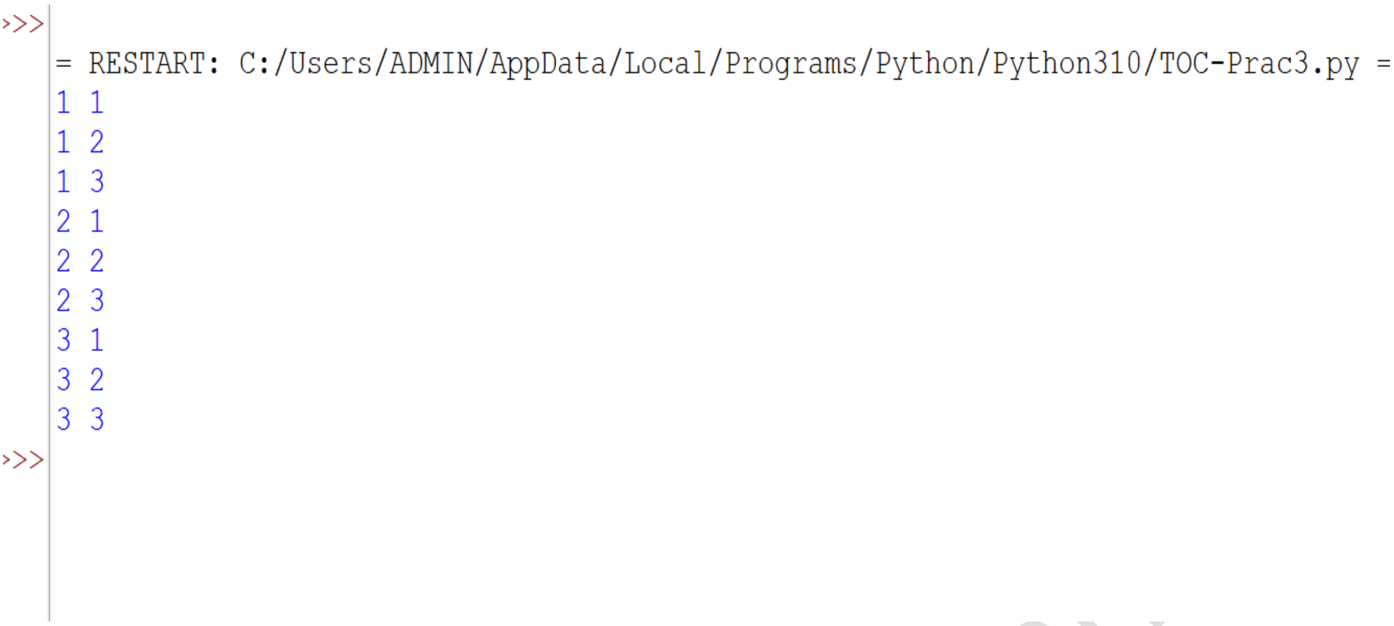
```
Python 3.10.0 (tags/v3.10.0:b494f59, Oct  4 2021, 19:00:18) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.

===================================== RESTART: C:/Users/shree/Documents/ree1.py ===================
searchObj.group() :  horses are taller than dogs
searchObj.group(1) :  horses
searchObj.group(2) :  taller
```

3} Write a program for generating derivation sequence / language for the given sequence of productions in Python.

```
# Python3 program of above approach

# A utility function that prints
# a given arr[] of length size#
def printArray(arr, size):
        for i in range(size):
                print(arr[i], end = " ")
        print()
        return

# This function returns 0 if there are
# no more sequences to be printed, otherwise
# modifies arr[] so that arr[] contains
# next sequence to be printed #
def getSuccessor(arr, k, n):

        # start from the rightmost side and
        # find the first number less than n
        p = k - 1
        while (arr[p] == n and 0 <= p < k):
                p -= 1

        # If all numbers are n in the array
        # then there is no successor, return 0
        if (p < 0):
                return 0

        # Update arr[] so that it contains successor
        arr[p] = arr[p] + 1
        i = p + 1
```

```
    while(i < k):
        arr[i] = 1
        i += 1
    return 1

# The main function that prints all sequences
# from 1, 1, ..1 to n, n, ..n
def printSequences(n, k):
    arr = [0] * k

    # Initialize the current sequence as
    # the first sequence to be printed #
    for i in range(k):
        arr[i] = 1

    # The loop breaks when there are
    # no more successors to be printed
    while(1):

        # Print the current sequence
        printArray(arr, k)

        # Update arr[] so that it contains
        # next sequence to be printed. And if
        # there are no more sequences then
        # break the loop
        if(getSuccessor(arr, k, n) == 0):
            break
    return

# Driver code
n = 3
k = 2
printSequences(n, k)
```

Output-



```
>>>
    = RESTART: C:/Users/ADMIN/AppData/Local/Programs/Python/Python310/TOC-Prac3.py =
    1 1
    1 2
    1 3
    2 1
    2 2
    2 3
    3 1
    3 2
    3 3
>>>
```

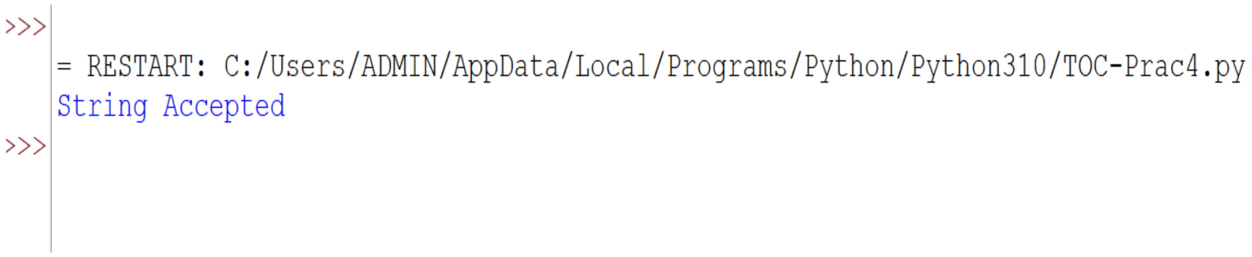
4} Design a program for creating machine that accepts three Consecutive one in Python.

```
# Python3 implementation of the
# DFA of permutation of three
# a's and three b's


# 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 Q

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 = "abaabb"

# call stateA
# to check the input
stateA(n)
```

Output-

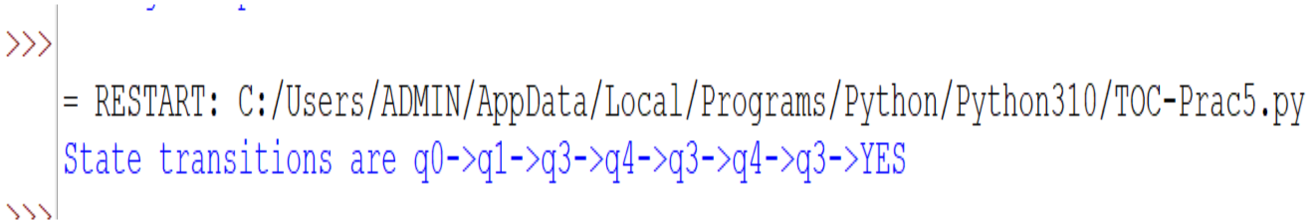


```
>>>
    = RESTART: C:/Users/ADMIN/AppData/Local/Programs/Python/Python310/TOC-Prac4.py
    String Accepted
>>>
```

5} Design a Program for creating machine that accepts the string always ending with 101 in python.

```
# Python3 Program to DFA that accepts string ending

# with 01 or 10.


# 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);
```

Output-



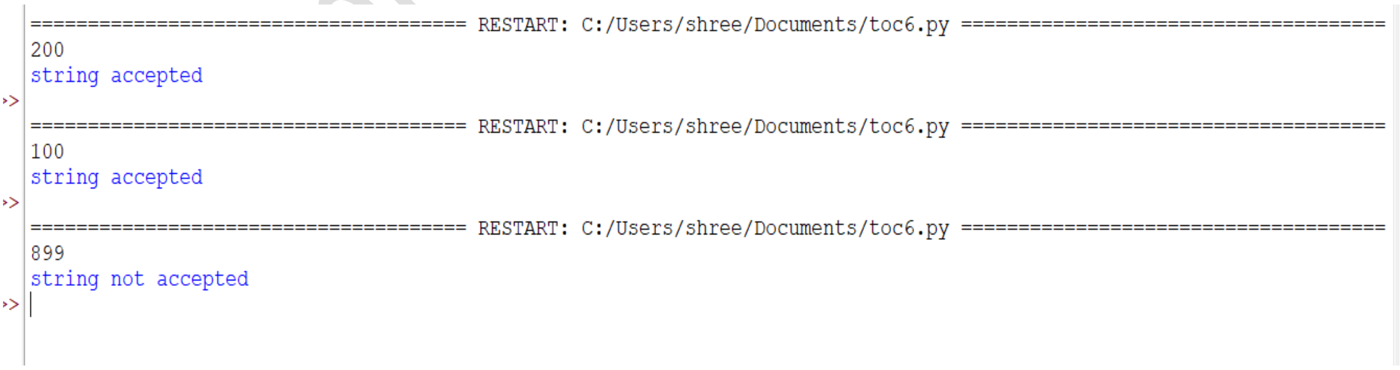
```
>>>
= RESTART: C:/Users/ADMIN/AppData/Local/Programs/Python/Python310/TOC-Prac5.py
State transitions are q0->q1->q3->q4->q3->q4->q3->YES
>>>
```

6} Design a program for accepting decimal number divisible by 2 in python.

```
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)
```

Output-

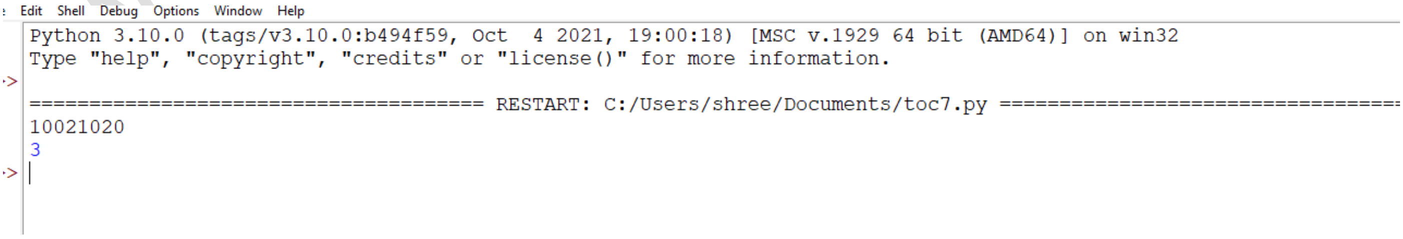


```
===================================== RESTART: C:/Users/shree/Documents/toc6.py ====================================
200
string accepted
>>
===================================== RESTART: C:/Users/shree/Documents/toc6.py ====================================
100
string accepted
>>
===================================== RESTART: C:/Users/shree/Documents/toc6.py ====================================
899
string not accepted
>>
```

7} Design a program for creating a machine which accepts string having equal no of 1's and 0's in Python.

```
# 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";
# Or
Str=input()  #take input from user at runtime

# Function call
print(getSubStringWithEqual012(Str));
```

Output-

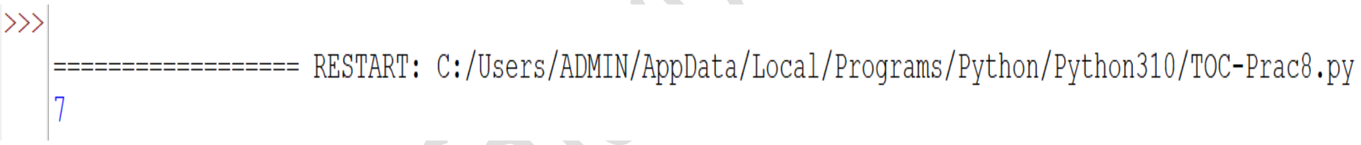


```
Python 3.10.0 (tags/v3.10.0:b494f59, Oct  4 2021, 19:00:18) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>
====================================== RESTART: C:/Users/shree/Documents/toc7.py ==================================
10021020
3
>
```

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

```
# Python3 implementation of the

# above approach


# Function to find the count

# of substrings with equal no.

# of consecutive 0's and 1's

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));
```

Output-



```
>>> 
================= RESTART: C:/Users/ADMIN/AppData/Local/Programs/Python/Python310/TOC-Prac8.py
7
```

Program 9: Design a PDA to accept WCWR where w is any string and WR is reverse of that string and C is a Special symbol.

==

Note: this question is difficult to implement practically as a program specially in python instead we have solved same type of question given below

You can implement above question in c or in c++ but it will be lengthy

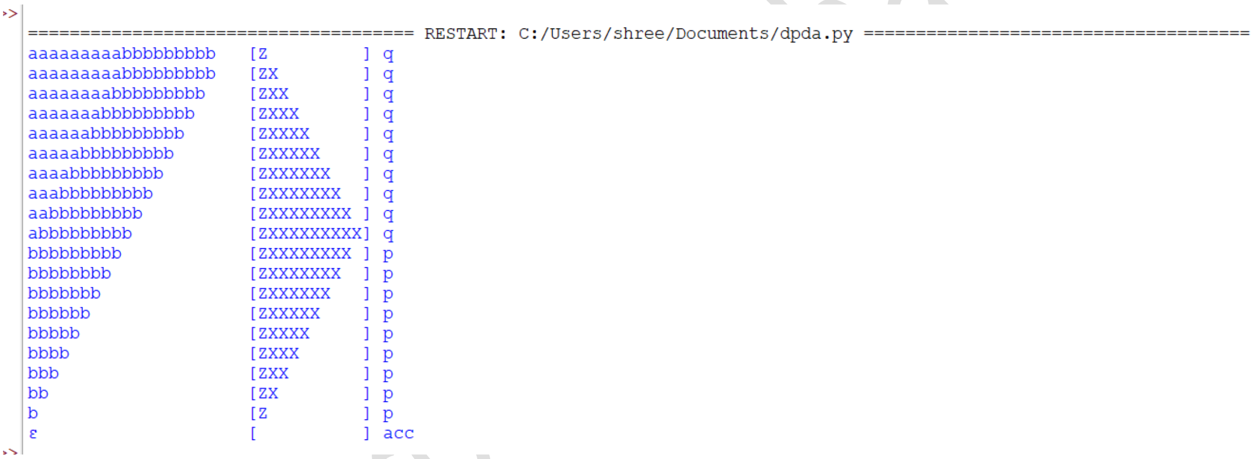
---

Alternative question

Deterministic Pushdown Automata for L = a^nb^n | n >=0) Python Program

```
class DPDA:


    def __init__(self, trf, input, state):
        self.head = 0
        self.trf = {}
        self.state = str(state)
        self.input = input
        self.trf = trf
        self.stack = ['Z']

    def step(self):

        a = self.input[self.head]
        s = self.stack.pop()
        state, ss = self.trf.get((self.state, a, s))
        if ss != 'ε':
            for s in ss[::-1]:
                self.stack.append(s)
        self.state = state
        print('{:20s} [{:10s}] {:5s}'.format(self.input[self.head:],
                       ''.join(self.stack), self.state))
        self.head += 1

    def run(self):

        print('{:20s} [{:10s}] {:5s}'.format(self.input[self.head:],
                              ''.join(self.stack), self.state))

        while self.head  < len(self.input):
            self.step()

        s = self.stack.pop()
        if self.trf.get((self.state, 'ε', s)):
            state, ss = self.trf.get((self.state, 'ε', s))
            self.state = state
            print('{:20s} [{:10s}] {:5s}'.format('ε',
                 ''.join(self.stack), self.state))


# run DPDA to accept the input string a^9b^9
DPDA({('q', 'a', 'Z'): ('q', 'XZ'),
     ('q', 'a', 'X'): ('q', 'XX'),
     ('q', 'b', 'X'): ('p', 'ε'),
     ('p', 'b', 'X'): ('p', 'ε'),
     ('p', 'ε', 'Z'): ('acc', 'Z'),
    },
    'aaaaaaaaabbbbbbbbb', 'q').run()
```

Output:



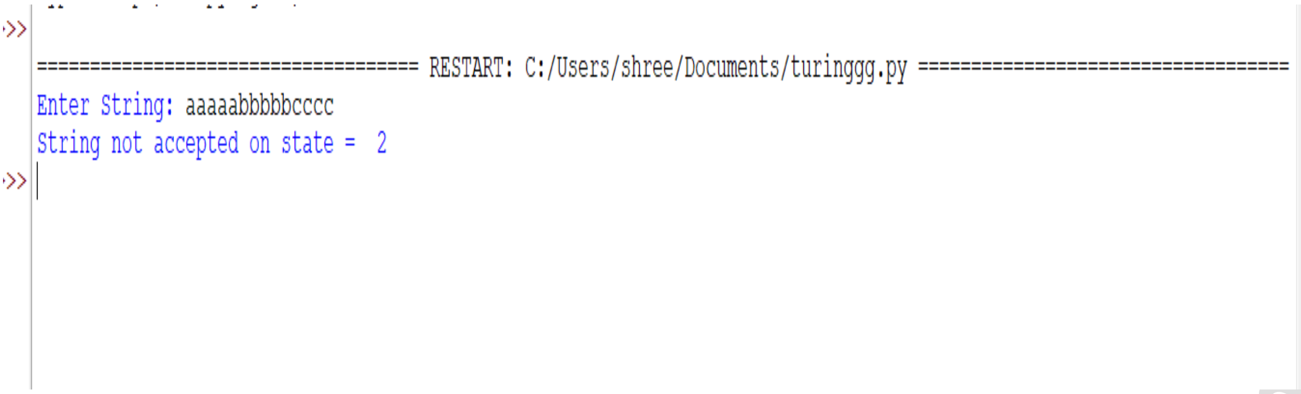
```
>>>
==================================== RESTART: C:/Users/shree/Documents/dpda.py ====================================
aaaaaaaaabbbbbbbbb   [Z         ] q
aaaaaaaaabbbbbbbbb   [ZX        ] q
aaaaaaaabbbbbbbbb    [ZXX       ] q
aaaaaaabbbbbbbbb     [ZXXX      ] q
aaaaaabbbbbbbbb      [ZXXXX     ] q
aaaaabbbbbbbbb       [ZXXXXX    ] q
aaaabbbbbbbbb        [ZXXXXXX   ] q
aaabbbbbbbbb         [ZXXXXXXX  ] q
aabbbbbbbbb          [ZXXXXXXXX ] q
abbbbbbbbb           [ZXXXXXXXXX] q
bbbbbbbbb            [ZXXXXXXXX ] p
bbbbbbbb             [ZXXXXXXX  ] p
bbbbbbb              [ZXXXXXX   ] p
bbbbbb               [ZXXXXX    ] p
bbbbb                [ZXXXX     ] p
bbbb                 [ZXXX      ] p
bbb                  [ZXX       ] p
bb                   [ZX        ] p
b                    [Z         ] p
ε                    [          ] acc
>>>
```

Program 10:

Design a Turing machine that’s accepts the following language an b n c
n where n>0

```
#function to perform action of states
def action(inp, rep, move):
    global tapehead
    if tape[tapehead] == inp:
        tape[tapehead] = rep
        if move == 'L':
            tapehead -= 1
        else:
            tapehead += 1
        return True
    return False

tape = ['B']*50
string = input("Enter String: ")
i = 5
tapehead = 5
for s in string: #loop to place string in tape
    tape[i] = s
    i += 1

state = 0
a, b, X, Z, U, V, R, L, B = 'a', 'b', 'X', 'Z', 'U', 'V', 'R', 'L', 'B'
oldtapehead = -1
accept = False
while(oldtapehead != tapehead): #if tapehead not moving that means terminate
Turing machine
    oldtapehead = tapehead

    if state == 0:
        if action(a, X, R):
            state = 1
        elif action(B, B, R):
            state = 10
        elif action(Z, Z, R):
            state = 7
        elif action(b, U, R):
            state = 4

    elif state == 1:
        if action(a, a, R):
            state = 1
        elif action(b, b, R):
            state = 2
        elif action(B, B, L):
            state = 11

    elif state == 2:
        if action(b, b, R) or action(Z, Z, R):
            state = 2
        elif action(a, Z, L):
            state = 3

    elif state == 3:
        if action(b, b, L) or action(Z, Z, L) or action(a, a, L):
            state = 3
        elif action(X, X, R):
            state = 0

    elif state == 4:
        if action(b, b, R):
            state = 4
        elif action(Z, Z, R):
            state = 5
        elif action(B, B, L):
            state = 15

    elif state == 5:
        if action(Z, Z, R) or action(V, V, R):
            state = 5
        elif action(b, V, L):
            state = 6
```

```
elif state == 6:
    if action(Z, Z, L) or action(V, V, L) or action(b, b, L):
        state = 6
    elif action(U, U, R):
        state = 0

elif state == 7:
    if action(Z, Z, R):
        state = 7
    elif action(V, V, R):
        state = 8

elif state == 8:
    if action(V, V, R):
        state = 8
    elif action(B, B, R):
        state = 9

elif state == 11:
    if action(a, a, L):
        state = 11
    elif action(X, X, R):
        state = 12

elif state == 12:
    if action(a, Z, R):
        state = 13

elif state == 13:
    if action(a, X, R):
        state = 12
        elif action(B, B, R):
            state = 14

    elif state == 15:
        if action(b, b, L):
            state = 15
        elif action(U, U, R):
            state = 16

    elif state == 16:
        if action(b, V, R):
            state = 17

    elif state == 17:
        if action(b, U, R):
            state = 16
        elif action(B, B, R):
            state = 18

    else:
        accept = True


if accept:
    print("String accepted on state = ", state)
else:
    print("String not accepted on state = ", state)
```



```
>>>
==================================== RESTART: C:/Users/shree/Documents/turinggg.py ====================================
Enter String: aaaaabbbbbcccc
String not accepted on state =  2
>>>
```

==

For more study material visit:

www.profajaypashankar.com

visit our youtube channel :

https://www.youtube.com/@ajaypashankar7

join our Telegram channel :

https://t.me/profajaypashankar