**THEORY OF**

**COMPUTATION**

**S.Y.B.Sc Computer**

**Science**

**(IV Semester)**

**For Academic Year**

**(2023-2024)**



**JNAN VIKAS MANDAL’S**

**PADMASHREE DR. R.T.DOSHI DEGREE COLLEGE OF**

**INFORMATION TECHNOLOGY**

**MOHANLAL RAICHAND MEHTA COLLEGE OF**

**COMMERCE DIWALIMAA DEGREE COLLEGE OF**

**SCIENCE**

**CERTIFICATE**

This is to certify that the Mr./Miss.

having roll number of S.Y.B.Sc.(CS) Semester-IV has completed the practical work in the subject of **THEORY OF COMPUTATION** during the Academic year 2023-2024 under the guidance of **Ms. Shakuntala Kulkarni.** being the partial requirement for the fulfilment of the curriculum of Degree of Bachelor of Science in Computer

Science, University of Mumbai.

**Place:**

**Date:**

Sign of Subject Incharge Sign of External Examiner

Sign of In charge / H.O.D

**INDEX**

|  |  |  |  |
| --- | --- | --- | --- |
| Sr.No. | Practicals | Date | Sign |
| 1 | Write a program for tokenization of given input in python. | 03/01/2024 |  |
| 2 | Write a program for generating regular  expressions for regular grammar in Python. | 03/01/2024 |  |
| 3 | Write a program for generating derivation  sequence / language for the given sequence of productions | 10/01/2024 |  |
| 4 | Design a program for creating machine that accepts three Consecutive one in Python. | 10/01/2024 |  |
| 5 | Design a program for accepting decimal number divisible by 2 in python. | 15/01/2024 |  |
| 6 | Design a program for creating a machine which accepts string having equal no of 1’s and 0’s in Python. | 15/01/2024 |  |
| 7 | Design a program for creating a machine which count number of 1’s and 0’s in agiven string in python. | 17/01/2024 |  |
| 8 | Design a Program for creating machine that accepts the string always ending with 101 in python. | 17/01/2024 |  |

SYCS Theory of Computation

**Practical 1**

**Write a program for tokenization of given input in python.**

In Python tokenization basically refers to splitting up a larger body of text into smaller lines, words or even creating words for anon-English language.

The split() method splits a string into a list.

**Code: -**

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: -**

["Let's", 'play', 'a', 'game,', 'Would', 'You', 'Rather!', "It's", 'simple,', 'you', 'have', 'to', 'pick', 'one', 'or', 'the', 'other.', 'L', "et's", 'get', 'started.', 'Would', 'you', 'rather', 'try', 'Vanilla', 'Ice', 'Cream', 'or', 'Chocolate', 'one?', 'Would', 'you', 'rather', 'be', 'a', 'b', 'ird', 'or', 'a', 'bat?', 'Would', 'you', 'rather', 'explore', 'space', 'or', 'the', 'ocean?', 'Would', 'you', 'rather', 'live', 'on', 'Mars', 'or', 'on', 'th', 'e', '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', 'g', 'uess,', 'you', 'pretty', 'much', 'not', 'understand', 'my', 'point,', 'neither', 'did', 'I,', 'at', 'first', 'place', 'and', 'that', 'led', 'me', 'to', 'a', 'Ba', 'd', 'Decision.']



|  |
| --- |
|  |

SYCS Theory of Computation

**Practical - 2**

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

**in Python.**

Python has a module named re to work with Regular Expression

The search () function searches the string for a match, and returns a Match object if there is a match.

[re.M](https://docs.python.org/3/library/re.html#re.M) is multi-line

re.I is Ignore Case

**Code: -**

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: -**

searchObj.group() : horses are taller than dogs

searchObj.group(1) : horses

searchObj.group(2) : taller



|  |
| --- |
|  |

SYCS Theory of Computation

**Practical 3**

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

# 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

SYCS Theory of Computation

# break the loop

if(getSuccessor(arr, k, n) == 0):

break

return

# Driver code

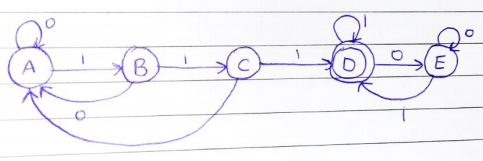
n = 3

k = 2

printSequences(n, k)

**Output:-**

**in Python.**



#StateA

def stateA(s,i):

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

if (i==len(s)):

print("String Not Accepted");

return;

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

stateA(s, i+1);

else:

stateB(s, i+1);

#StateB

def stateB(s,i):

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

if (i==len(s)):

print("String Not Accepted");

return;

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

stateA(s, i+1);

else:

stateC(s, i+1);

SYCS Theory of Computation

#StateC

def stateC(s,i):

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

if (i==len(s)):

print("String Not Accepted");

return;

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

stateA(s, i+1);

else:

stateD(s, i+1);

#StateD

def stateD(s,i):

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

if (i==len(s)):

print("String Accepted");

return;

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

stateE(s, i+1);

else:

stateD(s, i+1);

#StateE

def stateE(s,i):

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

if (i==len(s)):

print("String Accepted");

return;

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

stateE(s, i+1);

else:

SYCS Theory of Computation

stateD(s, i+1);

# Driver Code

if name == " main " :

s="001110";

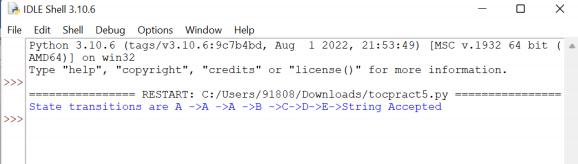
# all state transitions are printed.

# if string is accpetable, YES is printed.

# else NO is printed

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

stateA(s, 0);

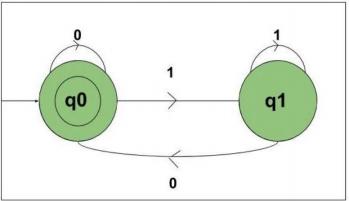


|  |
| --- |
|  |

SYCS Theory of Computation

**Practical - 5**

**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):

SYCS Theory of Computation

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())

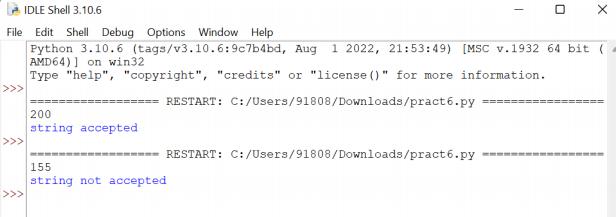
#converting number to binary

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

#call stateA

#to check the input

stateq0(n)



|  |
| --- |
|  |

SYCS Theory of Computation

**Practical – 6**

**Design a program for creating a machine which accepts string having equal**

**no of 2’s, 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):

SYCS

count += 1;

return count;

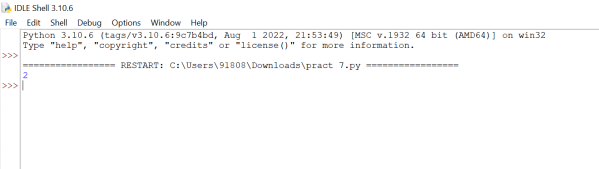
# Driver's code

Str = "0102010";

# Function call

print(getSubStringWithEqual012(Str));

Theory of Computation



SYCS Theory of Computation

**Practical – 7**

**Design a program for creating a machine which count number of 1’s and**

**0’s in a given string in python.**

s="001111001"

c1=0

c2=0

for i in s:

if i=='0':

c1=c1+1

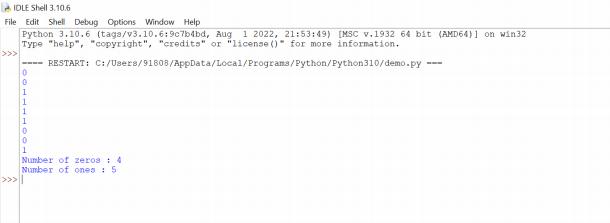
else:

c2=c2+1

print(i)

print("Number of zeros :",c1)

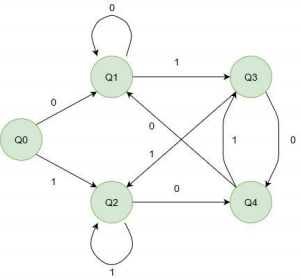
print("Number of ones :",c2)



SYCS Theory of Computation

**Practical - 8**

**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.

**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

SYCS Theory of Computation

**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



SYCS Theory of Computation

**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);