#practical 1

#write a program for tokenization of given input

string = "This is a sentence. Here is another one ."

tokens = string.split()

print(tokens)

#practical 2

# write a program for generating regular expression for regular grammer

import re

pattern = '^a...s$'

test\_string = 'abyss'

result = re.match(pattern, test\_string)

if result:

print("search successful")

else:

print("search unsuccessful")

#practical 3

# write a program for generating derivation sequence/language for given sequence of production

import random

def generate\_derivation(grammar, start\_symbol, max\_steps):

sequence, symbol = [], start\_symbol

for \_ in range(max\_steps):

if symbol not in grammar:

break

production = random.choice(grammar[symbol])

sequence.append(production)

symbol = production

return sequence

example\_grammar = {'5': ['AB', 'BC',], 'A': ['a'], 'B': ['b'], 'c': ['c']}

start\_symbol, max\_steps = 'A', 5

sequence = generate\_derivation(example\_grammar, start\_symbol, max\_steps)

print('Derivation sequence', sequence)

#practical 4

#Design a progam for creating machine that accepts three consecutive one

def has\_three\_consecutive\_ones(binary\_string):

return '111' in binary\_string

user\_input = input("senter a binary strinng: ")

if has\_three\_consecutive\_ones(user\_input):

print("the input string contains three consecutive '1's. ")

else:

print("the input string does not contain three consecutivr '1's. ")

#practical 5

#Design a program for for creting machine that accepts the string always ends with 101

def accepts\_string\_ending\_with\_101():

user\_input = input("Enter a string: ")

if user\_input.endswith("101"):

print("the string ends with 101")

else:

print("The string does not end with 101")

accepts\_string\_ending\_with\_101()

#practical 6

#Design a program for accepting decimal number divisible by 2

def check\_divisibility():

try:

decimal\_number = float(input("Enter a decimal number: "))

integer\_part = int(decimal\_number)

if integer\_part % 2 ==0:

print(f"{decimal\_number}is divisible by 2")

else:

print(f"{decimal\_number}is not divisible by 2")

except ValueError:

print("Invalid input, please enter a valid decimal number. ")

check\_divisibility()

#practical 7

#Design a program for creating a machine which accepts string having equal no. of 1's and 0's

def check\_equal(s):

count\_1s=s.count('1')

count\_0s=s.count('0')

if count\_1s==count\_0s:

return True

else:

return False

input\_string=input("Enter a string: ")

if check\_equal(input\_string):

print("The string has an equal number of 1's and 0's")

else:

print("The string doese not has equal number of 1's and 0's")

#practical 8

#Design a program for creating a machine which count no. of 1's and 0's in given string

def count\_numbers():

input\_string = input("Enter a string containing only '0's and '1's: ")

count\_0 = 0

count\_1 = 0

for char in input\_string:

if char in input\_string:

if char == '0':

count\_0 += 1

elif char == '1':

count\_1 += 1

else:

print("Invalid character in input string, please enter a string containing only '0's ")

return count\_0, count\_1

return count\_0, count\_1

count\_0, count\_1 = count\_numbers()

print("Number of '0's: ", count\_0)

print("Number of '1's: ", count\_1)

#practical 9

#Design a PDA to accept WCWR where w is any string and WR reverse of that string and C is is special symbol

def is\_wcwr(s): return len(s) % 2 == 1 and s [:len(s)//2] == s[:-len(s)//2-1:-1] and s[len(s)//2] == 'C'

input\_str = "abCba"

result = is\_wcwr(input\_str)

print(f' the string "{input\_str}" is {"in" if result else "not in"} the form WCWR ')

#practical 10

#Design a turing machine thats accepts the following language an b n c n where n>0

def simulate\_turing\_machine(input\_str):

tape, head, state = list(input\_str + '\_'), 0, 'q0'

while state != 'q\_accept' and state != 'q\_reject':

sym = tape[head]

if state == 'q0': tape[head], head, state = ('\_', head + 1,'q1') if sym == 'a' else('\_',0,'q\_reject')

elif state == 'q1': head, state = (head + 1, 'q1') if sym == 'a' else(head-1,'q2') if sym == 'b' else ('','q\_reject')

elif state == 'q2': head, state = (head - 1, 'q2') if sym == 'b' else(head+1,'q3') if sym == 'c' else ('','q\_reject')

elif state == 'q3': head, state = (head +1, 'q3') if sym == 'c' else('','q\_accept') if sym == '\_' else ('','q\_reject')

return state == 'q\_accept'

#exampple usage

input\_str = "aaabbbccc"

result = simulate\_turing\_machine(input\_str)

print(f'the string "{input\_str}" is {"accepted" if result else "rejected"} by the Turing machine.')