```
#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.')
```