ASSIGNMENT -1

Question 1:

Number game between user and computer. The user starts by entering either 1 or 2 or 3 digits starting from 1 sequentially. The computer can return either 1 or 2 or 3 next digits in sequence, starting from the max number played by the user. User enters the next 1 or 2 or 3 next digits in sequence, starting from the max number played by the computer. Whoever reaches 20 first wins the game.

Code 1:

```
import random
def computer_turn(current_number):
move = random.randint(1, 3)
next_numbers = list(range(current_number + 1, current_number + move + 1))
return next_numbers
def user_turn(current_number):
while True:
try:
user_input = input(f"Enter 1, 2, or 3 numbers starting from {current_number + 1}: ")
user_numbers = list(map(int, user_input.split()))
if len(user_numbers) in [1, 2, 3] and user_numbers[0] == current_number + 1 and user_numbers == list(range(user_numbers[0], user_numbers[0] + len(user_numbers))):
return user_numbers
else:
print("Invalid input. Please enter a valid sequence of 1, 2, or 3 consecutive numbers.")
except ValueError:
print("Invalid input. Please enter numbers only.")
def game():
current_number = 0
while current_number < 20:
user_numbers = user_turn(current_number)
```

```
current_number = user_numbers[-1]
print(f"You played: {user_numbers}")
if current_number >= 20:
print("Congratulations! You reached 20. You win!")
break
game()
```

Question 2:

Develop a function called ncr(n,r) which computes r-combinations of n-distinct object . use this function to print pascal triangle, where number of rows is the input

Code 2:

```
def factorial(num):
    if num == 0 or num == 1:
        return 1
    result = 1
    for i in range(2, num + 1):
        result *= i
    return result


def ncr(n, r):
    return factorial(n) // (factorial(r) * factorial(n - r))


def print_pascals_triangle(rows):
    for row in range(rows):
        line = []
        for col in range(row + 1):
            line.append(ncr(row, col))
        print(" " * (rows - row), "   ".join(map(str, line)))
```

```
num_rows = int(input("Enter the number of rows for Pascal's Triangle: "))
print_pascals_triangle(num_rows)
```

Question 3:

Read a list of n numbers during runtime. Write a Python program to print the repeated elements with frequency count in a list.

Code 3:

```
from collections import Counter
n = int(input("Enter the number of elements in the list: "))
numbers = []
print("Enter the numbers:")
for _ in range(n):
num = int(input())
numbers.append(num)
```

Question 4:

Develop a python code to read matric A of order 2X2 and Matrix B of order 2X2 from a file and perform the addition of Matrices A & B and Print the results.

Create a text file named matrices.txt

Code 4:

```
def read_matrix_from_file(filename):
    with open(filename, 'r') as file:
        lines = file.readlines()

    matrix_a = []
    for i in range(1, 3):
        row = list(map(int, lines[i].strip().split()))
        matrix_a.append(row)

    matrix_b = []
    for i in range(5, 7):
        row = list(map(int, lines[i].strip().split()))
        matrix_b.append(row)

    return matrix_a, matrix_b

def add_matrices(matrix_a, matrix_b):
    result = []
    for i in range(2):
        row = []
        for j in range(2):
            row.append(matrix_a[i][j] + matrix_b[i][j])
        result.append(row)
    return result

def print_matrix(matrix):
    for row in matrix:
        print(' '.join(map(str, row)))

filename = 'matrices.txt'
matrix_a, matrix_b = read_matrix_from_file(filename)
result_matrix = add_matrices(matrix_a, matrix_b)

print("Result of Matrix A + Matrix B:")
print_matrix(result_matrix)
```

Question 5:

Write a program that overloads the + operator so that it can add two objects of the class Fraction.

Fraction can be considered of the for P/Q where P is the numerator and Q is the denominator

Code 5:

```
class Fraction:
    def __init__(self, numerator, denominator):
        if denominator == 0:
            raise ValueError("Denominator cannot be zero")
        self.numerator = numerator
        self.denominator = denominator


    def __add__(self, other):
        if not isinstance(other, Fraction):
            return NotImplemented
        new_numerator = (self.numerator * other.denominator) + (other.numerator *
self.denominator)
        new_denominator = self.denominator * other.denominator
        return Fraction(new_numerator, new_denominator)


    def __str__(self):
        return f"{self.numerator}/{self.denominator}"


frac1 = Fraction(1, 2)
frac2 = Fraction(3, 4)
result_frac = frac1 + frac2
print(f"The result of adding {frac1} and {frac2} is {result_frac}")
```