Question 1

```
import random

def computer_play(max_num):
    # Computer plays 1, 2, or 3 digits in sequence from the max number
    next_numbers = random.randint(1, 3)
    moves = list(range(max_num + 1, max_num + next_numbers + 1))
    print(f"Computer played: {moves}")
    return moves

def user_play(max_num):
    while True:
        # User should enter 1, 2, or 3 digits in sequence
        user_input = input("Player: ")
        user_moves = list(map(int, user_input.split()))
        if len(user_moves) > 3 or len(user_moves) < 1:
            print("You must enter 1 to 3 numbers.")
            continue
        if user_moves != list(range(max_num + 1, max_num + 1 + len(user_moves))):
            print(f"Numbers should be in sequence starting from {max_num + 1}. Try again.")
            continue
        return user_moves

def play_game():
    print("Game Start! Reach 20 to win.")
    max_num = 0  # Game starts from 1
    while True:
        # User's turn
        user_moves = user_play(max_num)
        max_num = user_moves[-1]
        if max_num >= 20:
            print("Player Wins!!!")
            break

        # Computer's turn
        comp_moves = computer_play(max_num)
        max_num = comp_moves[-1]
        if max_num >= 20:
            print("Computer Wins!!!")
            break

# Start the game
play_game()
```

Question 2

```
import math

# Function to compute nCr (combinations)
def ncr(n, r):
    return math.factorial(n) // (math.factorial(r) * math.factorial(n - r))
```

```
# Function to print Pascal's Triangle
def print_pascals_triangle(rows):
    for n in range(rows):
        # Print leading spaces for alignment
        print(" " * (rows - n), end="")

        # Print the values in the row
        for r in range(n + 1):
            print(ncr(n, r), end=" ")

        print()  # Move to the next line after each row

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

Question 3

```
from collections import Counter

# Function to print the repeated elements with their frequencies
def print_frequencies(numbers):
    # Use Counter to get frequencies of each element
    frequency = Counter(numbers)

    # Iterate through the frequency dictionary and print the results
    for element, count in frequency.items():
        print(f"Element {element} has come {count} times")

# Input: List of numbers from the user
numbers = list(map(int, input("Enter a list of numbers separated by spaces: ").split()))

# Call the function to print frequencies
print_frequencies(numbers)
```

Question 4

```
def read_matrices_from_file(filename):
    with open(filename, 'r') as file:
        # Read the matrix data
        matrix_data = file.readlines()

        # Convert the matrix data into two 2x2 matrices
        matrix_a = []
        matrix_b = []

        # First two rows for matrix A
        matrix_a.append(list(map(int, matrix_data[0].split())))
        matrix_a.append(list(map(int, matrix_data[1].split())))
        # Next two rows for matrix B
        matrix_b.append(list(map(int, matrix_data[2].split())))
        matrix_b.append(list(map(int, matrix_data[3].split())))

    return matrix_a, matrix_b

def add_matrices(matrix_a, matrix_b):
    result = [[0, 0], [0, 0]]
    for i in range(2):
        for j in range(2):
            result[i][j] = matrix_a[i][j] + matrix_b[i][j]
    return result

def print_matrix(matrix, label):
    print(f"{label}:")
    for row in matrix:
        print(row)

# Read matrices from a file
filename = 'matrices.txt'  # Replace with your file name
matrix_a, matrix_b = read_matrices_from_file(filename)

# Perform matrix addition
result_matrix = add_matrices(matrix_a, matrix_b)

# Print the matrices and result
print_matrix(matrix_a, "Matrix A")
print_matrix(matrix_b, "Matrix B")
print_matrix(result_matrix, "Result Matrix")
```

Question 5

```
import math

class Fraction:
    def __init__(self, numerator, denominator):
        self.numerator = numerator
        self.denominator = denominator

    def __add__(self, other):
        # Adding two fractions: (P1/Q1) + (P2/Q2) = (P1*Q2 + P2*Q1) / (Q1*Q2)
        new_numerator = self.numerator * other.denominator + other.numerator * self.denominator
        new_denominator = self.denominator * other.denominator

        # Simplifying the fraction
        gcd = math.gcd(new_numerator, new_denominator)
        return Fraction(new_numerator // gcd, new_denominator // gcd)

    def __str__(self):
        # Return string representation of the fraction
        return f"{self.numerator}/{self.denominator}"
```

```
# Example usage
frac1 = Fraction(1, 2)  # 1/2
frac2 = Fraction(1, 3)  # 1/3

# Adding two fractions
result = frac1 + frac2

# Output the result
print(f"The sum of {frac1} and {frac2} is {result}")
```