```
# 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.
# Note:
#- the numbers should be in sequence starting from 1.
#- minimum number user or computer should pick is at least 1 digit in
sequence
#- maximum number user or computer can pick only 3 digits in sequence


# Number game between user and computer

def computer_move(last_num):
    target = 20
    move_count = (target - last_num - 1) % 4
    if move_count == 0:
        move_count = 1
    return list(range(last_num + 1, last_num + move_count + 1))

def number_game():
    print("Number Game: Reach 20 to Win!")
    last_num = 0

    while last_num < 20:
        # Player's turn
        player_input = input("Enter the next 1, 2, or 3 numbers in
sequence: ")
        player_numbers = list(map(int, player_input.split()))

        # Check if the player's numbers are valid
        if any(num <= last_num or num > last_num + 3 for num in
player_numbers) or len(player_numbers) > 3:
            print("Invalid move! Please enter 1 to 3 numbers in
sequence starting from the last number.")
            continue

        last_num = player_numbers[-1]

        # Check if the player wins
        if last_num >= 20:
            print("Player Wins!!!")
            break

        # Computer's turn
        computer_numbers = computer_move(last_num)
        print(f"Computer played: {computer_numbers}")
```

```
        last_num = computer_numbers[-1]

        # Check if the computer wins
        if last_num >= 20:
            print("Computer Wins!!!")
            break

# Start the game
number_game()
```

```
Number Game: Reach 20 to Win!
Enter the next 1, 2, or 3 numbers in sequence: 1 2
Computer played: [3]
Enter the next 1, 2, or 3 numbers in sequence: 5 6 7
Invalid move! Please enter 1 to 3 numbers in sequence starting from
the last number.
Enter the next 1, 2, or 3 numbers in sequence: 4 5 6
Computer played: [7]
Enter the next 1, 2, or 3 numbers in sequence: 8 9
Computer played: [10, 11]
Enter the next 1, 2, or 3 numbers in sequence: 12 13
Computer played: [14, 15]
Enter the next 1, 2, or 3 numbers in sequence: 14 15
Invalid move! Please enter 1 to 3 numbers in sequence starting from
the last number.
Enter the next 1, 2, or 3 numbers in sequence: 16 17
Computer played: [18, 19]
Enter the next 1, 2, or 3 numbers in sequence: 20
Player Wins!!!
```

```
# 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.

import math

def ncr(n, r):
    # Calculate r-combinations of n
    return math.factorial(n) // (math.factorial(r) * math.factorial(n
- r))

def print_pascal_triangle(rows):
    for n in range(rows):
        # Generate each row using ncr function
        row = [ncr(n, r) for r in range(n + 1)]
        # Print the row, centered for a triangle shape
        print(" " * (rows - n), end="")
        print(" ".join(map(str, row)))

# Input: number of rows for Pascal's triangle
```

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

```
Enter the number of rows for Pascal's Triangle: 12
           1
          1 1
         1 2 1
        1 3 3 1
       1 4 6 4 1
      1 5 10 10 5 1
     1 6 15 20 15 6 1
    1 7 21 35 35 21 7 1
   1 8 28 56 70 56 28 8 1
  1 9 36 84 126 126 84 36 9 1
 1 10 45 120 210 252 210 120 45 10 1
1 11 55 165 330 462 462 330 165 55 11 1
```

```
# 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.

from collections import Counter

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

# Count the frequency of each element in the list
frequency_count = Counter(numbers)

# Print only the repeated elements with their frequency count
print("Repeated elements with their frequency count:")
for num, count in frequency_count.items():
    if count > 1:
        print(f"{num}: {count}")
```

```
Enter numbers separated by spaces: 1 2 3 4 5 6 7 8 2 3 4 5 6 7 2  3 4
Repeated elements with their frequency count:
2: 3
3: 3
4: 3
5: 2
6: 2
7: 2
```

```
# 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.

def read_matrix(file):
    """Read a 2x2 matrix from the file."""
    matrix = []
```

```
    for _ in range(2):  # Read 2 rows for a 2x2 matrix
        line = file.readline()
        row = list(map(int, line.split()))
        matrix.append(row)
    return matrix

def add_matrices(matrix_a, matrix_b):
    """Add two 2x2 matrices."""
    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):
    """Print the matrix."""
    for row in matrix:
        print(' '.join(map(str, row)))

# Read matrices from the file
with open('/content/matrices.txt', 'r') as file:
    matrix_a = read_matrix(file)
    matrix_b = read_matrix(file)

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

# Print the result
print("Matrix A:")
print_matrix(matrix_a)
print("\nMatrix B:")
print_matrix(matrix_b)
print("\nResult of A + B:")
print_matrix(result_matrix)
```

```
Matrix A:
1 2
3 4

Matrix B:
5 6
7 8

Result of A + B:
6 8
10 12
```

```python
# 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

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

    def simplify(self):
        """Simplify the fraction to its simplest form."""
        def gcd(a, b):
            while b:
                a, b = b, a % b
            return a

        common = gcd(abs(self.numerator), abs(self.denominator))
        self.numerator //= common
        self.denominator //= common

    def __add__(self, other):
        """Overload the + operator to add two fractions."""
        if isinstance(other, Fraction):
            new_numerator = (self.numerator * other.denominator) +
(other.numerator * self.denominator)
            new_denominator = self.denominator * other.denominator
            return Fraction(new_numerator, new_denominator)
        return NotImplemented

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

def main():
    # Input for the first fraction
    num1 = int(input("Enter numerator for the first fraction: "))
    denom1 = int(input("Enter denominator for the first fraction: "))
    fraction1 = Fraction(num1, denom1)

    # Input for the second fraction
    num2 = int(input("Enter numerator for the second fraction: "))
    denom2 = int(input("Enter denominator for the second fraction: "))
    fraction2 = Fraction(num2, denom2)

    # Add the two fractions
    result = fraction1 + fraction2
```

```
    # Print the result
    print(f"{fraction1} + {fraction2} = {result}")

if __name__ == "__main__":
    main()
```

```
Enter numerator for the first fraction: 2
Enter denominator for the first fraction: 4
Enter numerator for the second fraction: 6
Enter denominator for the second fraction: 7
1/2 + 6/7 = 19/14
```