In [6]:
```
import numpy as np
import random
```

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 Example 1: Player: 1 2 Computer played: [3, 4] Player: 5 6 7 Computer played: [8, 9] Player: 10 Computer played: [11, 12, 13] Player: 14 15 Computer played: [16, 17, 18] Player: 19 20 Player Wins!!!

Example 2: Player: 1 Computer played: [2, 3] Player: 4 5 Computer played: [6, 7, 8] Player: 9 10 Computer played: [11] Player: 12 Computer played: [13] Player: 14 15 Computer played: [16] Player: 17 18 Computer played: [19, 20] Computer Wins!!!

In [30]:
```
def number_game():
    max_num = 0
    goal = 20

    print("Let's start the game! Reach 20 first to win.")

    while max_num < goal:
        # User move
        while True:
            try:
                user_input = input(f"Enter 1, 2, or 3 numbers in sequence, starting from {max_num + 1}: ")
                user_numbers = list(map(int, user_input.split(',')))

                # Validate user input (1 to 3 numbers, in correct sequence)
                if len(user_numbers) in [1, 2, 3] and user_numbers == list(range(max_num + 1, max_num + 1 + len(user_
                    max_num = user_numbers[-1]
```

```
                    print(f"You played: {user_numbers}")
                    break
                else:
                    print("Invalid input. Please enter a valid sequence of 1, 2, or 3 numbers.")
            except ValueError:
                print("Invalid input. Please enter numbers only.")

        if max_num >= goal:
            print("You reached 20! You win!")
            break

        # Computer move
        computer_choice = random.randint(1, 3)
        computer_numbers = list(range(max_num + 1, max_num + 1 + computer_choice))
        max_num = computer_numbers[-1]
        print(f"Computer played: {computer_numbers}")

        if max_num >= goal:
            print("Computer reached 20! Computer wins!")
            break

# Start the game
number_game()
```

```
Let's start the game! Reach 20 first to win.
You played: [1, 2, 3]
Computer played: [4, 5]
You played: [6]
Computer played: [7]
You played: [8, 9]
Computer played: [10, 11]
Invalid input. Please enter a valid sequence of 1, 2, or 3 numbers.
You played: [12, 13, 14]
Computer played: [15]
You played: [16, 17]
Computer played: [18, 19, 20]
Computer reached 20! Computer 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

In [13]:
```
def pascals_triangle(rows):
    def factorial(x):
        result = 1
        for i in range(2, x + 1):
            result *= i
        return result

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

    # Generate Pascal's Triangle
    for n in range(rows):
        # Print spaces for formatting
        print(' ' * (rows - n), end='')

        # Print values for current row
        for r in range(n + 1):
            print(ncr(n, r), end=' ')
        print()  # Newline after each row

# Example usage: input number of rows
num_rows = int(input("Enter the number of rows for Pascal's triangle: "))
pascals_triangle(num_rows)
```

```
     1
    1 1
   1 2 1
  1 3 3 1
 1 4 6 4 1
1 5 10 10 5 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.

Example :
Input:- [2,1,2,3,4,5,1,3,6,2,3,4]

Output:-

- Element 2 has come 3 times
- Element 1 has come 2 times
- Element 3 has come 2 times
- Element 4 has come 2 times
- Element 1 has come 1 times
- Element 6 has come 1 times

In [16]:
```python
from collections import Counter

def find_repeated_elements():
    # Input the list size
    n = input("Enter the number of elements in the list: ")

    # Input the list elements
    print(f"Enter {n} numbers:")
    numbers = list(map(int, n.split(',')))

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

    # Filter and print elements that are repeated
    #print("\nRepeated elements with frequency count:")
    for element, count in frequency.items():
        if count >= 1:
            print(f"Element {element} has come {count} times")

# Example usage
find_repeated_elements()
```

```
Enter 2,1,2,3,4,5,1,3,6,2,3,4 numbers:
Element 2 has come 3 times
Element 1 has come 2 times
Element 3 has come 3 times
Element 4 has come 2 times
Element 5 has come 1 times
Element 6 has come 1 times
```

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.

In [29]:
```
def read_matrices_from_file(filename):
    with open(filename, 'r') as file:
        # Read all lines from the file
        lines = file.readlines()

        # Read matrix A (first two lines)
        A = [[int(num) for num in lines[0].split()],
             [int(num) for num in lines[1].split()]]

        # Read matrix B (next two lines)
        B = [[int(num) for num in lines[2].split()],
             [int(num) for num in lines[3].split()]]

    return A, B

def add_matrices(A, B):
    # Adding corresponding elements of matrix A and matrix B
    result = [[A[i][j] + B[i][j] for j in range(2)] for i in range(2)]
    return result

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

# Example usage
filename = "matrix.txt"  # The file should contain 4 lines representing the 2x2 matrices A and B
A, B = read_matrices_from_file(filename)

# Perform the matrix addition
result = add_matrices(A, B)

# Print the matrices and result
print_matrix(A, "Matrix A")
```

```
print_matrix(B, "Matrix B")
print_matrix(result, "Result (A + B)")
```

```
Matrix A:
[1, 2]
[3, 4]
Matrix B:
[5, 6]
[7, 8]
Result (A + B):
[6, 8]
[10, 12]
```

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

In [27]:
```
import math

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):
        # Add two fractions using the formula: (P1 * Q2 + P2 * Q1) / (Q1 * Q2)
        new_numerator = (self.numerator * other.denominator) + (other.numerator * self.denominator)
        new_denominator = self.denominator * other.denominator

        # Simplify the result using the GCD
        gcd = math.gcd(new_numerator, new_denominator)
        return Fraction(new_numerator // gcd, new_denominator // gcd)

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

In [28]:
```python
# Example usage
fraction1 = Fraction(1, 2)  # 1/2
fraction2 = Fraction(1, 3)  # 1/3

result = fraction1 + fraction2  # Add fractions
print(f"Result of addition: {result}")  # Output: 5/6
```

```
Result of addition: 5/6
```

In [ ]: