ASSIGNMENT 01

Name: Kodurupaka Rajesh

Hall Ticket Number: 2406DGAL153

Mail id: rajesh.kluniversity@gmail.com

Mobile number: 9133245888

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:

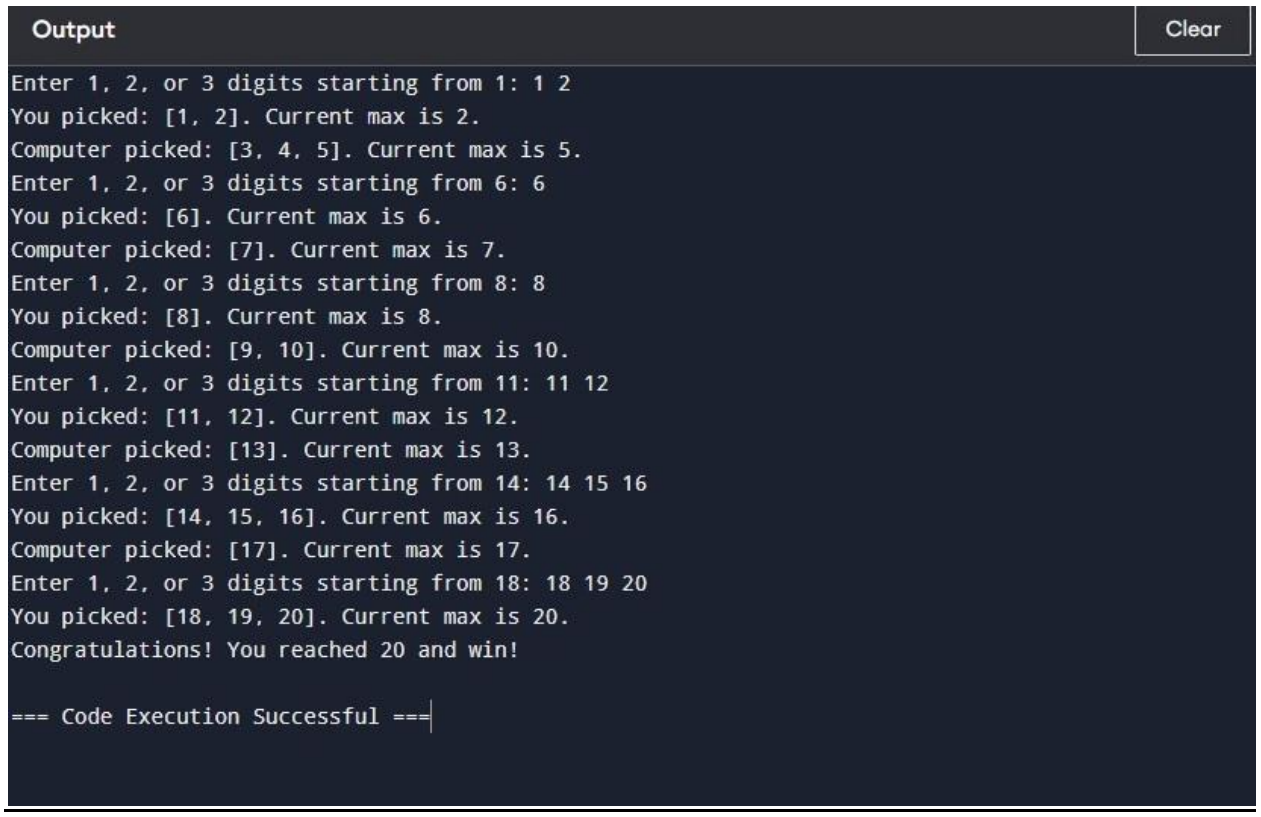
```
Player: 1
Computer played: [2, 3]
Player: 4 5
Computer played: [6, 7, 8]
Player: 9 10
Computer played: [11]
Player: 12
Computer played: [13]
Player: 14 15
Computer played: [16]
Player: 17 18
Computer played: [19, 20]
Computer Wins!!!
```

ANSWER:

```
import random


def get_user_input(current_max):
    while True:
        try:
            user_input = input(f"Enter 1, 2, or 3 digits starting from {current_max + 1}: ")
            numbers = list(map(int, user_input.split()))
            if all(num == current_max + i + 1 for i, num in enumerate(numbers)) and 1 <=
len(numbers) <= 3:
                return numbers
            else:
                print(f"Invalid input. Please enter 1 to 3 sequential numbers starting from
{current_max + 1}.")
```

```
        except ValueError:
            print("Invalid input. Please enter valid integers.")


def get_computer_input(current_max):
    count = random.randint(1, 3)  # Computer picks 1 to 3 digits
    return [current_max + i + 1 for i in range(count)]


def play_game():
    current_max = 0

    while current_max < 20:
        # User's turn
        user_numbers = get_user_input(current_max)
        current_max += len(user_numbers)
        print(f"You picked: {user_numbers}. Current max is {current_max}.")
        if current_max >= 20:
            print("Congratulations! You reached 20 and win!")
            break

        # Computer's turn
        computer_numbers = get_computer_input(current_max)
        current_max += len(computer_numbers)
        print(f"Computer picked: {computer_numbers}. Current max is {current_max}.")
        if current_max >= 20:
            print("Computer reached 20. You lose!")


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

ANSWER 01 OUTPUT:



```
Enter 1, 2, or 3 digits starting from 1: 1 2
You picked: [1, 2]. Current max is 2.
Computer picked: [3, 4, 5]. Current max is 5.
Enter 1, 2, or 3 digits starting from 6: 6
You picked: [6]. Current max is 6.
Computer picked: [7]. Current max is 7.
Enter 1, 2, or 3 digits starting from 8: 8
You picked: [8]. Current max is 8.
Computer picked: [9, 10]. Current max is 10.
Enter 1, 2, or 3 digits starting from 11: 11 12
You picked: [11, 12]. Current max is 12.
Computer picked: [13]. Current max is 13.
Enter 1, 2, or 3 digits starting from 14: 14 15 16
You picked: [14, 15, 16]. Current max is 16.
Computer picked: [17]. Current max is 17.
Enter 1, 2, or 3 digits starting from 18: 18 19 20
You picked: [18, 19, 20]. Current max is 20.
Congratulations! You reached 20 and win!

=== Code Execution Successful ===
```

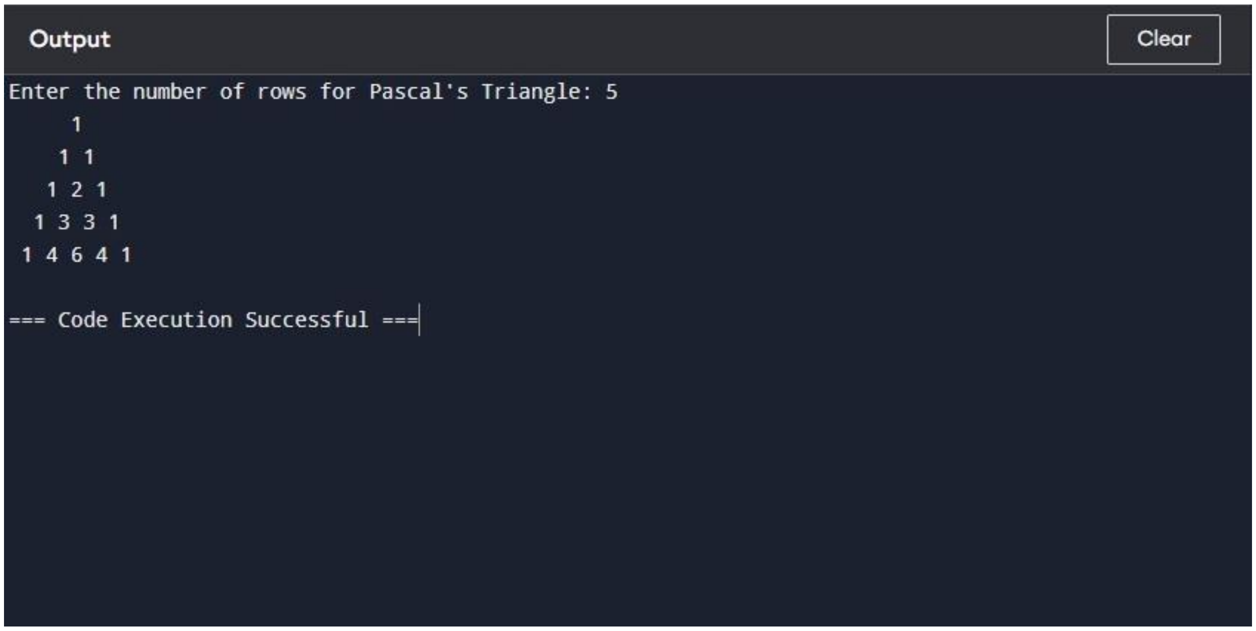
Question 2:

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

ANSWER:

```
def ncr(n, r):
    if r > n or r < 0:
        return 0
    if r == 0 or r == n:
        return 1
    # Calculate nCr using the formula n! / (r! * (n - r)!)
    num = 1
    denom = 1
    for i in range(r):
        num *= (n - i)
        denom *= (i + 1)
    return num // denom

def print_pascals_triangle(rows):
    for i in range(rows):
        # Print leading spaces for formatting
        print(' ' * (rows - i), end='')
        for j in range(i + 1):
            print(ncr(i, j), end=' ')
        print()

# Main function to get user input and print Pascal's Triangle
def main():
    rows = int(input("Enter the number of rows for Pascal's Triangle: "))
    print_pascals_triangle(rows)

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

<u>ANSWER 02 OUTPUT:</u>



```
Enter the number of rows for Pascal's Triangle: 5
     1
    1 1
   1 2 1
  1 3 3 1
 1 4 6 4 1

=== Code Execution Successful ===
```

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]

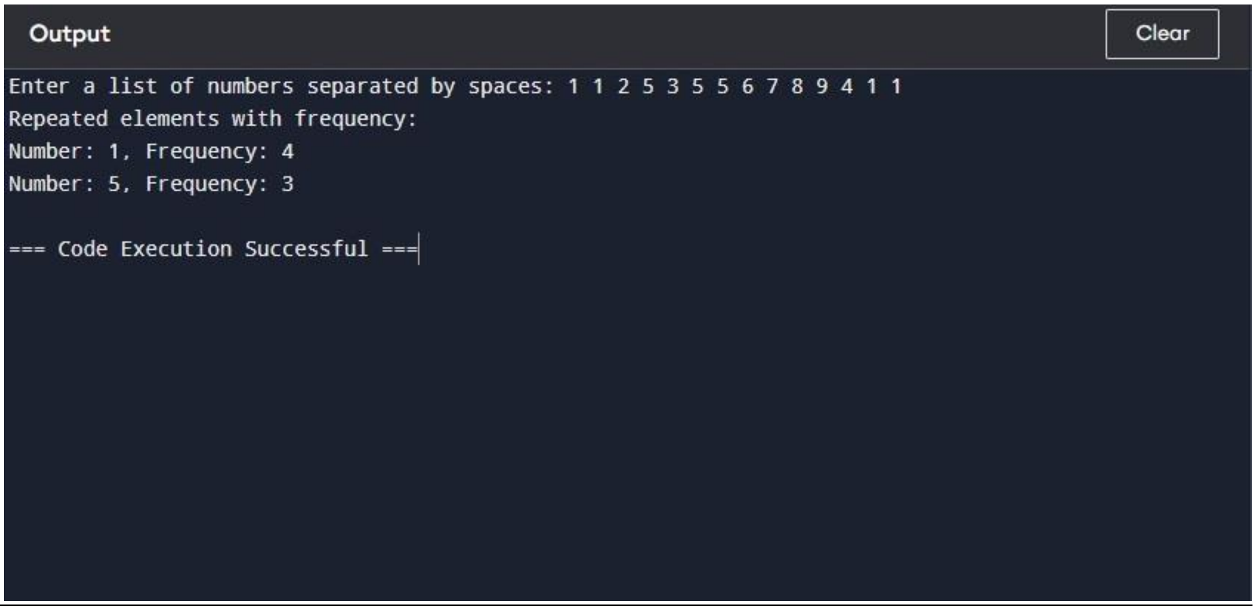
Output:-

```
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
```

ANSWER:

```
from collections import Counter


def main():
    # Read a list of numbers from user input
    numbers = input("Enter a list of numbers separated by spaces: ").split()

    # Convert the input strings to integers
    numbers = list(map(int, numbers))

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

    # Print the repeated elements with their frequency count
    print("Repeated elements with frequency:")
    for number, count in frequency.items():
        if count > 1:
            print(f"Number: {number}, Frequency: {count}")


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

ANSWER 03 OUTPUT:



```
Enter a list of numbers separated by spaces: 1 1 2 5 3 5 5 6 7 8 9 4 1 1
Repeated elements with frequency:
Number: 1, Frequency: 4
Number: 5, Frequency: 3

=== Code Execution Successful ===
```

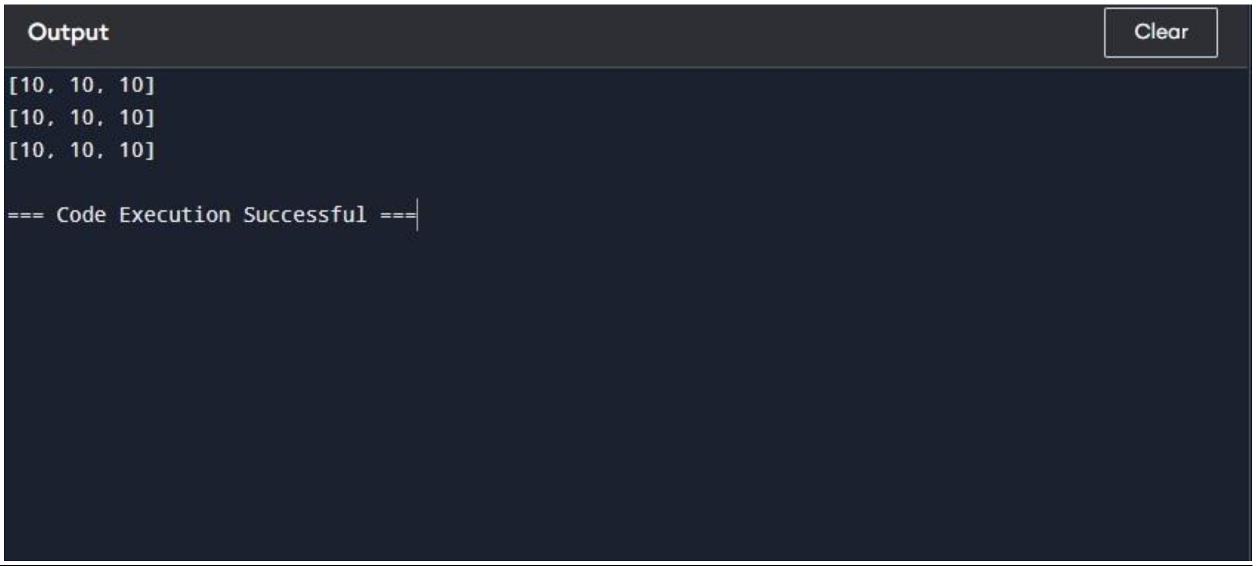
Question 4:-

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.

ANSWER:

```
# Program to add two matrices using nested loop
X = [[1, 2, 3],
    [4, 5, 6],
    [7, 8, 9]]

Y = [[9, 8, 7],
    [6, 5, 4],
    [3, 2, 1]]

result = [[0, 0, 0],
         [0, 0, 0],
         [0, 0, 0]]

# iterate through rows
for i in range(len(X)):
   # iterate through columns
   for j in range(len(X[0])):
       result[i][j] = X[i][j] + Y[i][j]

for r in result:
   print(r)
```

ANSWER 04 OUTPUT:



```
[10, 10, 10]
[10, 10, 10]
[10, 10, 10]

=== Code Execution Successful ===
```

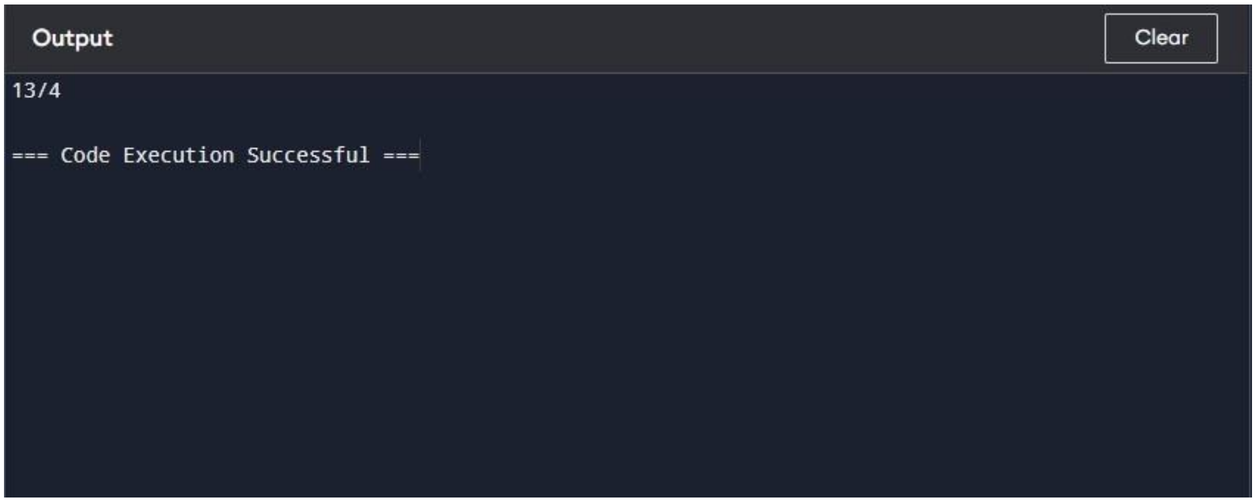
Question 5:-

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

ANSWER:

```
from math import gcd


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

# Example usage:
fraction1 = Fraction(7, 4)
fraction2 = Fraction(9, 6)

result = fraction1 + fraction2
print(result)  # Output: 13/4
```

ANSWER 05 OUTPUT:



```
13/4

=== Code Execution Successful ===
```