Name:\_\_\_\_\_ID:\_\_\_

## **Object-Oriented Programming Lab #2**

6/12/2023

## Introduction to C++

1. Given the following program:

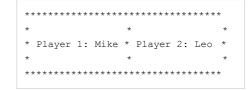
```
#include <iostream>
#include <string>

int main()
{
    std::cout << "Please enter P1 name: ";
    std::string p1_name;
    std::cin >> p1_name;

    std::cout << "Please enter P2 name: ";
    std::string p2_name;
    std::cin >> p2_name;

    std::cout << "Player 1: " << p1_name << std::endl;
    std::cout << "Player 2: " << p2_name << std::endl;
    return 0;
}</pre>
```

- 1.1) What will the above program do if you type two names (for example, "John Smith")
- **1.2)** Change std::cin >> p1\_name; with std::getline(std::cin, p1\_name); Recompile and run program, see the result
- **1.3)** Change the program so that it draws frame around the name for both players like the example output shown on the right:



#### TA Check Point 1

**1.4)** Change the program so that it draws frame around the name for both players like shown below:

Output (for <b>2.3a</b> )	Output (for <b>2.3b</b> )	Output (for <b>2.3c</b> )
*****	++	+======+
* *	T I	
* Player 1: Mike *	Player 1: Mike	Player 1: Mike
* *	T I	
******	++	++
* *	T I	
* Player 2: Leo *	Player 2: Leo	Player 2: Leo
* *	T I	
******	++	+======+

#### TA Check Point 2

2. Write programs to print patterns with varying sizes.

All programs **must take** the pattern size from user and **must not** print trailing spaces before the end of each line.

**2.1)** The program should print a triangle pattern like shown below:

Output Size = 0 (0 line)	Output Size = 1	Output Size = 2
	*	*
		**
Output Size = 3	Output Size = 4	Output Size = 5
*	*	*
**	**	**
***	***	***
	***	***
		****

# $\textbf{2.2)} \quad \text{The program should print an arrow pattern like shown below:} \\$

Output Size = 0 (0 line)	Output Size = 1	Output Size = 2
	*	*
		**
		*
Output Size = 3	Output Size = 4	Output Size = 5
*	*	*
**	**	**
***	***	***
**	***	***
*	***	****
	**	***
	*	***
		**
		*

### TA Check Point 3

- 3. The Monte Carlo method for estimating  $\pi$  is based on the probability of randomly generated points falling inside a unit circle. Here's how the concept is applied:
  - 1. Random Points Generation: Generate N random points (xi, yi), where each xi and yi is within the range [-1, 1] inclusive. This effectively places the points within a 2x2 square centered at the origin (0,0).
  - 2. Unit Circle Check: A point (xi, yi) lies inside the unit circle (radius 1) if the distance from the origin,  $d = \sqrt{x^2 + y^2}$ , is less than or equal to 1.
  - 3. Probability and  $\pi$  Estimation: The probability of a point lying inside the circle is approximately the ratio of the area of the circle to the area of the square. Since the area of the circle is  $\pi r^2$  ( $\pi$  in this case since r=1) and the area of the square is 4 (2x2 square), this ratio is  $\pi/4$ . Thus, by multiplying the probability by 4, we get an approximation of  $\pi$ .

```
#include <random>
#include <iostream>
#include <iomanip>
#include <cmath>
class Rand double {
public:
    Rand_double(double low, double high) : dist(low, high) {
        std::random_device rd;
        re.seed(rd());
    double operator()() {
        return dist(re);
private:
    std::default_random_engine re;
    std::uniform_real_distribution<double> dist;
int main() {
    const double rnd_min = -1.0, rnd_max = 1.0;
    Rand_double rnd(rnd_min, rnd_max);
    const int N = ; // Number of points to generate
    int points_inside = 0;
    for (int i = 0; i < N; ++i) {
        double x = rnd();
        double y = rnd();
        if (x * x + y * y \le 1) {
            ++points_inside;
    std::cout << std::fixed << std::setprecision(3);</pre>
    std::cout << "Estimated Pi: " << pi_estimate << std::endl;</pre>
    return 0;
```

#### **Code Explanation:**

Class Definition - Rand\_double: This class is designed to generate random double values within a specified range [-1, 1]. It initializes a uniform real distribution and a random engine. The random engine generates random numbers, which are then scaled to the specified range by the distribution.

Main Function - Input and Setup: The program prompts the user to enter the number of points (N) to be used for the estimation. It initializes the Rand\_double class to generate random points within the required range.

Estimation Loop: For each of the N points, the program generates a random (x, y) pair. It checks if each point lies inside the unit circle  $(x^2 + y^2 \le 1)$ . The count of points inside the circle is maintained.

Calculation of  $\pi$ : After all points are processed, the program estimates  $\pi$  as  $4 \times$  (number of points inside the circle) / N. This value is printed as the estimated value of  $\pi$ .

- **3.1)** Correct 6he code
- **3.2)** Estimate Pi using N = 100, record the approximation
- 3.3) Estimate Pi using N = 1,000, record the approximation
- **3.4)** Estimate Pi using N = 10,00, record the approximation

TA Check Point 4