Lab 4

Goals-

Identify requirements for a program using inheritance

Develop a plan to test your program against requirements and verify the results

The following lists a Dice class that simulates rolling a die with a different number of sides. The default is a standard die with six sides. The rollTwoDice function simulates rolling two dice objects and returns the sum of their values. The srand function requires including cstdlib .

```
class Dice
{
public:
  Dice();
  Dice( int numSides);
  virtual int rollDice() const;
protected:
  int numSides;
};


Dice::Dice()
{
  numSides = 6;
  srand(time(NULL)); // Seeds random number generator
}


Dice::Dice(int numSides)
{
  this->numSides = numSides;
  srand(time(NULL)); // Seeds random number generator
}


int Dice::rollDice() const
{
  return (rand() % numSides) + 1;
}


// Take two dice objects, roll them, and return the sum
int rollTwoDice(const Dice& die1, const Dice& die2)
{
  return die1.rollDice() + die2.rollDice();
}
```

Write a main function that creates two Dice objects with a number of sides of your choosing. Invoke the rollTwoDice function in a loop that iterates ten times and verify that the functions are working as expected.

Next create your own class, LoadedDice , that is derived from Dice . Add a default constructor and a constructor that takes the number of sides as input. Override the rollDice function in LoadedDice so that with a 50% chance the function returns the largest number possible (i.e., numSides), otherwise it returns what Dice's rollDice function returns.

Test your class by replacing the Dice objects in main with LoadedDice objects.
You should not need to change anything else. There should be many more dice rolls with the highest possible value. Polymorphism results in LoadedDice's rollDice function to be invoked instead of Dice's rollDice function inside rollTwoDice.

Do a more formal analysis of your change. Calculate descriptive statistics (i.e. mean, median, mode, and standard deviation) for regular and loaded dice. Roll a die 100 times, or more. Collect the results. Calculate the statistics. You can save the output into a file (remember to comma or tab separate the numbers to make it easier to import into a tool). You can find and use functions to calculate the statistics after all the dice have been rolled. Try each pair (regular and loaded) with dice of a different number of sides.

Does the change in statistics make sense? Is it what you expected? If you need a refresher remember that math is fun is your friend! Seriously, go to mathisfun.com. ☺ Submit a short document that shows the numbers and your brief analysis.

HINT: Think about the problem. Develop your design. Write the design down. Implement. Test. Revise (if necessary). In this case, the statistical analysis is part of the testing.