1. Using the `For` command, write a program that adds the cubes of the first 100 positive integers. I.e., your program should compute $1^3 + 2^3 + \cdots + 100^3$. (The answer is 25,502,500.)

```plaintext
runningsum = 0;
For[i = 1, i <= 100, i++,
    runningsum += i^3;
];
Print[runningsum];
```

2. Using the `For` command, write a program that computes the number of positive integers that evenly divide 4,000,000.

```plaintext
count = 0;
number = 4000000;
For[i = 1, i <= number, i++,
    If[Mod[number, i] == 0, count++;
];
Print[count];
```

3. Using the `For` command, write a program that computes the sum of all the positive integers that evenly divide 4,000,000.

```plaintext
runningsum = 0;
number = 4000000;
For[i = 1, i <= number, i++,
    If[Mod[number, i] == 0, runningsum += i];
];
Print[runningsum];
```

4. (a) Write a program that flips a fair coin 10 times, then prints 1 if there were 7 or more heads, and prints 0 if not.

(b) Write a program that flips 10 fair coins 10000 times, and reports the number of times that 7 or more of the 10 flips were heads.

```plaintext
(* part (b) *)
howManyYeses = 0;
For[i = 1, i <= 10000, i++,
    numberOfHeads = 0;
    For[j = 1, j <= 10, j++,
        x = RandomInteger[];
        If[x == 1, numberOfHeads++]
    ];
    If[numberOfHeads >= 7, howManyYeses++]
];
Print[howManyYeses];
```

5. (a) Write a program that rolls a 6-sided die 3 times. If the maximum of the rolls is 5 or greater, print “yes”. Otherwise, print “no”.

(b) Write a program that approximates the probability that the maximum of 3 rolls of a six-sided die is 5 or greater.
6. (a) Write a program that rolls a 20-sided die 14 times. If the maximum of the rolls is 18 or greater, print “yes”. Otherwise, print “no”.

(b) Write a program that approximates the probability that the maximum of 14 rolls of a 20-sided die is 18 or greater.

Use the program from the previous problem with sidesOnDie, rollsPerTrial, and desiredMax set appropriately.

7. (a) Write a program that rolls an $x$-sided die $y$ times. If the maximum of the rolls is $z$ or greater, print “yes”. Otherwise, print “no”. Note: the first three lines of your program should look like this:

   ```plaintext
   x=[some number];
y=[some number];
z=[some number];
```

(b) Write a program that approximates the probability that the maximum of $y$ rolls of an $x$-sided die is $z$ or greater.

   Same as above.

8. Carl and Lankewicz are going to play checkers, best three out of five games. Suppose Lankewicz’s probability of winning a given game is .62.

(a) Write a program that plays five games of checkers between Carl and Lankewicz, and prints the name of the winner of the 5-game series.

(b) Write a program that plays 1000 five-game series of checkers and prints the percentage of the series that each player won.

(c) Same question, but this time they’re playing four-out-of-seven instead of three-out-of-five.

(d) Same question, but this time they’re playing five-out-of-nine.

(e) Same question, but this time they’re playing 53-out-of-105.

   ```plaintext
   LankProb = .62;
   bigNumber = 1000;
   gamesPerSeries = 21;
   ```
HowManySeriesWonByLank = 0;
HowManySeriesWonByCarl = 0;
For[i = 1, i <= bigNumber, i++,
    LankWinsSoFarInThisSeries = 0;
    CarlWinsSoFarInThisSeries = 0;
    For[j = 1, j <= gamesPerSeries, j++,
        x = RandomReal[];
        If[x < LankProb,
            LankWinsSoFarInThisSeries++,
            CarlWinsSoFarInThisSeries++
        ];
    ];
    If[LankWinsSoFarInThisSeries > CarlWinsSoFarInThisSeries,
        HowManySeriesWonByLank++,
        HowManySeriesWonByCarl++
    ];
];
Print["Lank percentage = ", 100*N[HowManySeriesWonByLank/bigNumber]];
Print["Carl percentage = ", 100*N[HowManySeriesWonByCarl/bigNumber]];

9. Here is a game called “fizzbuzz.”
   (a) Start counting with 1.
   (b) When you get to a multiple of 3, say “fizz.”
   (c) On multiples of 5, say “buzz.”
   (d) On numbers that are multiples of both 3 and 5, say “fizzbuzz.”
   (e) On all other numbers, say the number.

   So it goes like this: 1, 2, fizz, 4, buzz, fizz, 7, 8, fizz, buzz, 11, fizz, 13, 14, fizzbuzz, 16, …

   Write a program that takes a positive integer \( n \) as input, and then prints the list of fizzes, buzzes, and numbers up to \( n \).

   Let’s cancel this one.

10. Consider the following game:
   (a) start with your favorite positive integer.
   (b) if it’s even, divide it by 2.
   (c) if it’s odd, multiply it by 3 and add 1.
   (d) repeat until you get to 1.

   For example, if you started with 6, it would go like this: 3, 10, 5, 16, 8, 4, 2, 1.

   Write a program whose input is a positive integer, and whose output is the number of steps required before the process terminates. As an example, if \( n = 6 \), the output would be 8.

   Let’s cancel this one too.