Laboratory 06A

CS-102

Spring 2022

Laboratory 6A

- A prime number is a number that is only evenly divisible by itself and 1. For example:
  - The number 5 is prime because it can only be evenly divided by 1 and 5. The number 6, however, is not prime because it can be divided by 1, 2, 3 and 6.
- Write a function called isprime(), which takes an integer as an argument and returns true if the argument is a prime number, or false otherwise.
- Tips:
  - Program6-15 shows you how you can use the Boolean function, isEven(val), where the test is to see if the number, val, is odd or even. Your job is to alter the function, isEven(val), in that program to see if the number is Prime or not. Call your new function, isprime(val)
  - Recall that the % operator divides one number by another, and returns the remainder of the division. In an expression such as num1 % num2, the % operator will return 0 if num1 is evenly divisible by num2.
  - What you will do is to test the number, val, to see if it is divisible by 2, 3, …, val-1. If it is not, it is prime. If it is divisible by 2 or 3 or … val-1, then it is not Prime.

Laboratory 6A Part 1

- Call your program, *YourName*-Lab6A-1.cpp .
- When you have gotten your program working:
  - If you are doing Lab06A synchronously, call over the instructor so that you can be given proper credit for your work.
  - If you are doing Lab06A asynchronously, please submit your program to Canvas.

Laboratory 6A-2

- Use your new function, isprime(), in a program that will display all the Prime numbers which are less than 200. You can call this new program: *YourName*-Lab6A-2.cpp .
- When you have gotten your program for part 2 working:
  - If you are doing Lab06A synchronously, call over the instructor so that you can be given proper credit for your work.
  - If you are doing Lab06A asynchronously, please submit your program to Canvas.

Laboratory 6A-3

- Now rewrite your Lab6A-2.cpp solution so that instead of displaying all the prime numbers up to the number input, instead you simply count the number and then display that number. Call your new solution Lab6A-3.
  - Note that there is no need to change your function isprime(n).
- Test your Prime counter, as well as time the length of time that it takes to count the number of primes up to N for N = 200,000.
- Record the number of primes that you found, as well as the length of time it took to generate the result. Put your results in a text file called *YourName-Lab6A-3.*
- When you have gotten your program for part 3 working:
  - If you are doing Lab06A synchronously, call over the instructor so that you can be given proper credit for your work.
  - If you are doing Lab06A asynchronously, please submit your program to Canvas.

Laboratory 6A-4

- The function, isprime(), that you used in the first part of this lab is likely not as efficient as it could be. The following is a version called isPrime() that is far more efficient, and hence will allow us to compute much larger numbers of prime numbers during the short span of a lab period.
- Now rewrite your Lab6A-3.cpp solution so that instead of using the version of isprime() that you have used up till now, you use the version isPrime() shown on the next page. Call your new solution Lab6A-4.
  - Note that there is no need to change the main() part of the program, only the function.
- Test your Prime counter, as well as time the length of time that it takes to count the number of primes up to N for N = 200,000.
- Record the number of primes that you found, as well as the length of time it took to generate the result. Put your results in a text file called *YourName-Lab6A-3.*
- When you have gotten your program for part 4 working:
  - If you are doing Lab06A synchronously, call over the instructor so that you can be given proper credit for your work.
  - If you are doing Lab06A asynchronously, please submit your program to Canvas.

```
//  An efficient version of the function isPrime(int n)
#include <iostream>
using namespace std;
bool isPrime(int n)
{
  if (n == 1)                    // 1 is not a prime
      return false;
  if (n == 2)                    // 2 is a prime
      return true;
  if (n % 2 == 0)                // No other even number is a prime
      return false;
  // Try finding a number that divides evenly into n
  int k = 3;                     // No need to divide by 2 since n is odd
  while (k * k <= n)             // Only need to try divisors up to sqrt(n)
  {
      if (n % k == 0)            // n is not a prime since it is divisible by k
            return false;
      k = k + 2;                 // Try next odd number
  }
  return true;                   // No divisor found. Therefore, n is a prime
}
```

An efficient version of the function isPrime(n)

Laboratory 6A-5

- Now combine this function, isPrime(), with a function main(), that you will write, which will count and print out the number of prime numbers in each **ennead*** (single digit, two digit, three digit, …, eight digit).
- You may write one program which gives the result for all 8 **enneads** or you may run the program for Laboratory6A-4.cpp eight times.
  - Note: If you choose to run the program from part 4, eight times, then you will need to subtract off the number of primes from each ennead less than the ennead you are currently computing.
  - For example: #primesInEnnead8 = total#primesUpTo10^8 - #primesEnnead7 - #primesEnnead6 - #primesEnnead5 - #primesEnnead4 - #primesEnnead3 - #primesEnnead2 - #primesEnnead1
- Call the resulting program, *YourName*Lab6A-5.cpp.
- If you are doing this Lab synchronously, show your results (the number of primes in each **ennead**) for all 8 **ennead**s.
- If you are doing this Lab asynchronously, submit your program & results to Canvas.

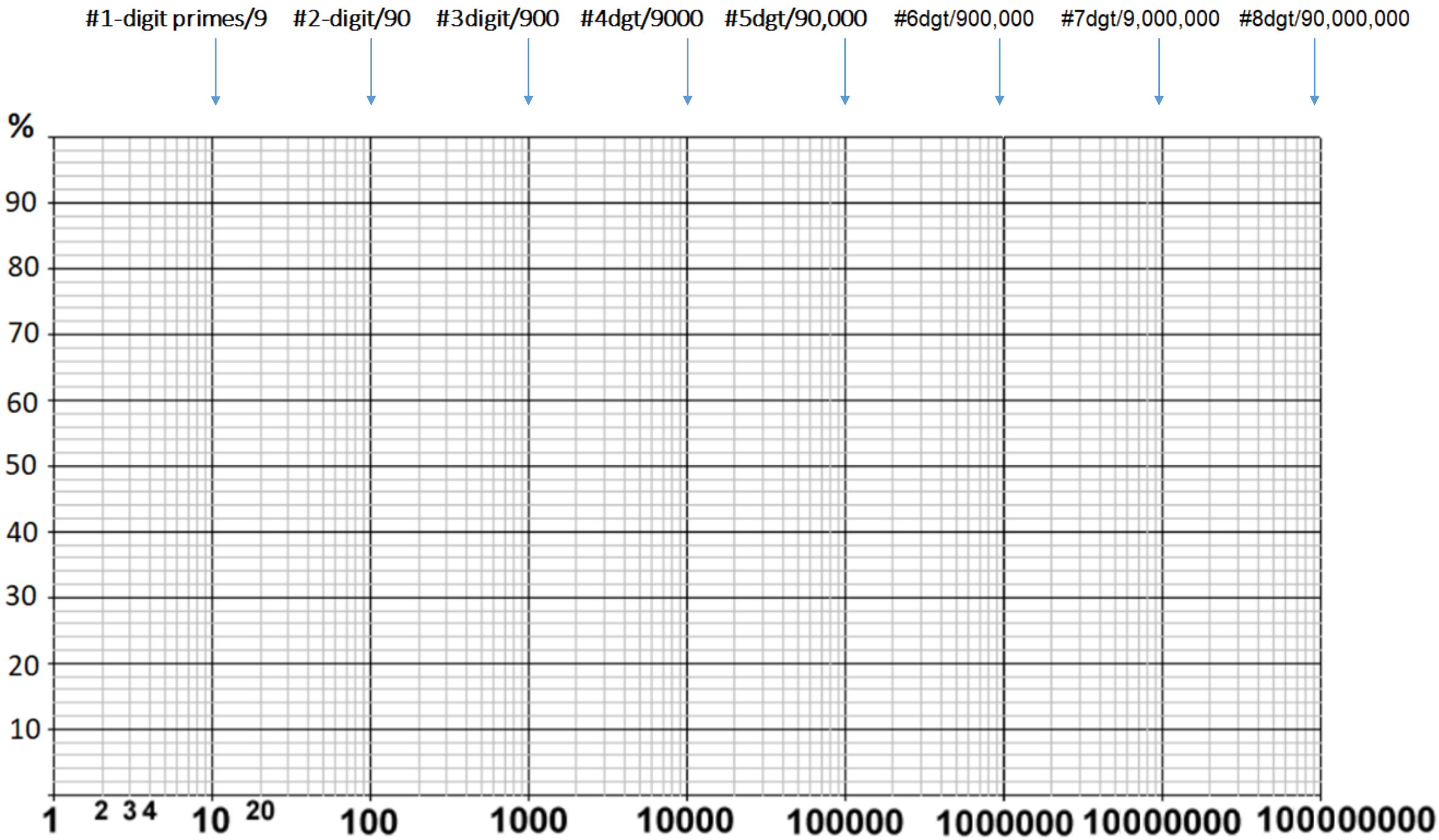
***Note the use of the word **ennead** (group of 9), rather than the word **decade** (group of 10). Notice that there are only 90 two digit numbers, 900 three digit numbers, 9000 four digit numbers, 90,000 five digit numbers, etc. This distinction will become important when you compute and graph the percentage of primes.

Laboratory 6A-6

- Use your new function, isPrime(), in a program that will count all the prime numbers less than or equal to the number that you type in. You can call this program: *YourName*-Lab6A-6.cpp .
- Record the number of Prime Numbers per power of 10 <= N, where N is:
  - 10 – Record the number of prime numbers less than 10.
  - 100 – Record the number of prime numbers less than 100 but more than 10.
  - 1,000 – Record the number of prime numbers less than 1,000 but more than 100.
  - 10,000 – Record the number of prime numbers less than 10,000 but more than 1,000.
  - 100,000 – Record the number of prime numbers less than 100,000 but more than 10,000.
  - 1,000,000 - Record the number of prime numbers less than 1,000,000 but more than 100,000.
  - 10,000,000 - Record the number of prime numbers less than 10,000,000 but more than 1,000,000.
  - 100,000,000 - Record the number of prime numbers less than 100,000,000 but more than 10,000,000.

Laboratory 6A-6

- Graph the percentage of prime numbers, in each of these 8 groups, (2-9, 10-99, 100 – 999, 1000 – 9999, 10000 – 99999, 100000 – 999999, 1000000 – 9999999, 10000000 - 99999999).
  - What you are graphing is: # Prime Numbers(per **ennead**)/N where N is the number of numbers per **ennead**. (e.g. N would be 90 for two digit numbers, 900 for 3 digit, etc.)
- Do you think the number of prime numbers (per **ennead**) ever goes to zero as N goes to infinity?
- You will be using the graph shown on the next page to plot your results.


#1-digit primes/9
#2-digit/90
#3digit/900
#4dgt/9000
#5dgt/90,000
#6dgt/900,000
#7dgt/9,000,000
#8dgt/90,000,000
%
90
80
70
60
50
40
30
20
10
1
2
3
4
10
20
100
1000
10000
100000
1000000
10000000
100000000

Laboratory 6A-6

- Fill out the attached graph, Lab06A-Graph, using paint.exe (on a PC) or paintbrush (on a Macintosh).
- Make a table of the exact number of primes that you found in each **ennead** of your computation.
- From that table compute the % of primes found in each **ennead**.
- Graph the % of primes on the attached graph and submit that graph.
- If you are doing this Lab synchronously, show your graph to the instructor for assessment.
- If you are doing this Lab asynchronously, submit your graph to Canvas.

Program 6-15

```
1  // This program uses a function that returns true or false.
2  #include <iostream>
3  using namespace std;
4
5  // Function prototype
6  bool isEven(int);
7
8  int main()
9  {
10    int val;
11
12    // Get a number from the user.
13    cout << "Enter an integer and I will tell you ";
14    cout << "if it is even or odd: ";
15    cin >> val;
16
17    // Indicate whether it is even or odd.
18    if (isEven(val))
19       cout << val << " is even.\n";
20    else
21       cout << val << " is odd.\n";
22    return 0;
23 }
24
```

Returning a Boolean Value in Program 6-15

(Program Continues)

Returning a Boolean Value in Program 6-15

```
25  //*************************************************************
26  // Definition of function isEven. This function accepts an    *
27  // integer argument and tests it to be even or odd. The function *
28  // returns true if the argument is even or false if the argument *
29  // is odd. The return value is a bool.                        *
30  //*************************************************************
31
32  bool isEven(int number)
33  {
34     bool status = true;
35
36     if (number % 2 != 0)
37        status = false; // The number is even if there is no remainder.
38     return status;
39  }
```

Program Output with Example Input Shown in Bold

```
Enter an integer and I will tell you if it is even or odd: 5 [Enter]
5 is odd.
```