

Directions: SHOW ALL YOUR WORK. REMEMBER THAT PROGRAM SEGMENTS ARE TO BE WRITTEN IN JAVA.

Notes:

- Assume that the classes listed in the Quick Reference found in the Appendix have been imported where appropriate.
- Unless otherwise noted in the question, assume that parameters in method calls are not null and that methods are called only when their preconditions are satisfied.
- In writing solutions for each question, you may use any of the accessible methods that are listed in classes defined in that question. Writing significant amounts of code that can be replaced by a call to one of these methods may not receive full credit.

1. In mathematics, a prime number is a positive integer that is only divisible by 1 and itself. The first several prime numbers are 2, 3, 5, 7, 11, 13, 17 (1 is not considered to be prime). Below is the outline of a class that will generate prime numbers in ascending order. PrimeGenerator works by keeping tracking of the current prime number, and when asked, will give that value out and then store the next prime number. In this way, PrimeGenerator stores the next prime in the sequence of prime numbers **BEFORE** it is asked to give out that value.

```
public class PrimeGenerator
{
    /**
     * The next prime in the sequence of prime numbers.
     * 2 is considered the first prime number.
     */
    private int nextPrime = 2;

    /**
     * creates a default prime generator
     * the result of calling nextPrime() will be 2
     */
    public PrimeGenerator()
    { /* implementation not shown */ }

    /**
     * returns whether the number p is prime
     */
    private boolean isPrime(int p)
    { /* implementation not shown */ }

    /**
     * returns the next prime after p
     * Precondition: p is a prime
     */
    private int nextPrimeAfter(int p)
    { /* to be implemented in part (a) */ }

    /**
     * returns the next prime in the sequence of prime numbers
     * Precondition: nextPrime is the prime desired
     * Postcondition: nextPrime is changed to be the next prime
     * @return the nextPrime in the sequence of prime numbers
     */
    public int nextPrime()
    { /* to be implemented in part (b) */ }
}
```

(a) Write the method `nextPrimeAfter`. This method returns the next prime after prime `p`.

Complete method `nextPrimeAfter` below.

```
/**
 *   returns the next prime after p
 *   Precondition: p is a prime
 */
private int nextPrimeAfter(int p)
```

(b) Write method `nextPrime`. This method returns the `nextPrime` in the sequence of prime numbers. For example:

```
PrimeGenerator pg = new PrimeGenerator();
System.out.println(pg.nextPrime()); //2
System.out.println(pg.nextPrime()); //3
System.out.println(pg.nextPrime()); //5
System.out.println(pg.nextPrime()); //7
```

Complete method `nextPrime` below.

```
/**
 * returns the next prime in the sequence of prime numbers
 * Precondition: nextPrime is the prime desired
 * Poscondition: nextPrime is changed to be the next prime (e.g. 2 -> 3)
 * @return the nextPrime in the sequence of prime numbers
 */
public int nextPrime()
```

(c) In a completely separate class, we'd like to now find how many prime numbers are under 100,000. Finish this main method that will print out the number of primes under 100,000.

```
public class Driver
{
    public static void main(String[] args)
    {

}
}
```