Question – 1:

(The MyInteger class) Design a class named MyInteger. The class contains:

- An int data field named value that stores the int value represented by this object.
- A constructor that creates a MyInteger object for the specified int value.
- A getter method that returns the int value.
- The methods isEven(), isOdd(), and isPrime() that return true if the value in this object is even, odd, or prime, respectively.
- The static methods isEven(int), isOdd(int), and isPrime(int) that return true if the value in this object is even, odd, or prime, respectively.
- The static methods isEven(MyInteger), isOdd(MyInteger), and isPrime(MyInteger) that return true if the specified value is even, odd, or prime, respectively.
- The methods equals(int) and equals(MyInteger) that return true if the value in this object is equal to the specified value.
- A static method parseInt(char[]) that converts an array of numeric characters to an int value.
- A static method parseInt(String) that converts a string into an int value.

Draw the UML diagram for the class and then implement the class. Write a client program that tests all methods in the class.

Question -2:

(The MyPoint class) Design a class named MyPoint to represent a point with x- and y-coordinates. The class contains:

- The data fields x and y that represent the coordinates with getter methods.
- A no-arg constructor that creates a point (0, 0).
- A constructor that constructs a point with specified coordinates.
- A method named distance that returns the distance from this point to a specified point of the MyPoint type.
- A method named distance that returns the distance from this point to another point with specified x- and y-coordinates.

Draw the UML diagram for the class and then implement the class. Write a test program that creates the two points (0, 0) and (10, 30.5) and displays the distance between them.
Question-3:

(Displaying the prime factors) Write a program that prompts the user to enter a positive integer and displays all its smallest factors in decreasing order. For example, if the integer is 120, the smallest factors are displayed as 5, 3, 2, 2, 2. Use the StackOfIntegers class to store the factors (e.g., 2, 2, 2, 3, 5) and retrieve and display them in reverse order.

Question-4:

(The Queue class) Section 10.6 gives a class for Stack. Design a class named Queue for storing integers. Like a stack, a queue holds elements. In a stack, the elements are retrieved in a last-in first-out fashion. In a queue, the elements are retrieved in a first-in first-out fashion. The class contains:

- An int[] data field named elements that stores the int values in the queue.
- A data field named size that stores the number of elements in the queue.
- A constructor that creates a Queue object with default capacity 8.
- The method enqueue(int v) that adds v into the queue.

- The method dequeue() that removes and returns the element from the queue.
- The method empty() that returns true if the queue is empty.
- The method getSize() that returns the size of the queue.

Draw an UML diagram for the class. Implement the class with the initial array size set to 8. The array size will be doubled once the number of the elements exceeds the size. After an element is removed from the beginning of the array, you need to shift all elements in the array one position the left. Write a test program that adds 20 numbers from 1 to 20 into the queue and removes these numbers and displays them.

Question-5:

(Implement the StringBuilder class) The StringBuilder class is provided in the Java library. Provide your own implementation for the following methods (name the new class MyStringBuilder1):

```java
public MyStringBuilder1(String s);
public MyStringBuilder1 append(MyStringBuilder1 s);
public MyStringBuilder1 append(int i);
public int length();
public char charAt(int index);
public MyStringBuilder1 toLowerCase();
public MyStringBuilder1 substring(int begin, int end);
public String toString();
```