Write a generic method to count the number of elements in a collection that have a specific property (for example, odd integers, prime numbers, palindromes).

Will the following class compile? If not, why?

1. public final class Algorithm {
2. public static <T> T max(T x, T y) {
3. return x > y ? x : y;
4. }
5. }

Write a generic method to exchange the positions of two different elements in an array.

If the compiler erases all type parameters at compile time, why should you use generics?

What is the following class converted to after type erasure?

1. public class Pair<K, V> {
2. public Pair(K key, V value) {
3. this.key = key;
4. this.value = value;
5. }
6. public K getKey() { return key; }
7. public V getValue() { return value; }
8. public void setKey(K key) { this.key = key; }
9. public void setValue(V value) { this.value = value; }
10. private K key;
11. private V value;
12. }

What is the following method converted to after type erasure?

1. public static <T extends Comparable<T>>
2. int findFirstGreaterThan(T[] at, T elem) {
3. // ...
4. }

Will the following method compile? If not, why?

1. public static void print(List<? extends Number> list) {
2. for (Number n : list)
3. System.out.print(n + " ");
4. System.out.println();
5. }

Write a generic method to find the maximal element in the range [begin, end) of a list.

Will the following class compile? If not, why?

1. public class Singleton<T> {
2. public static T getInstance() {
3. if (instance == null)
4. instance = new Singleton<T>();
5. return instance;
6. }
7. private static T instance = null;
8. }

Given the following classes:

1. class Shape { /\* ... \*/ }
2. class Circle extends Shape { /\* ... \*/ }
3. class Rectangle extends Shape { /\* ... \*/ }
4. class Node<T> { /\* ... \*/ }

Will the following code compile? If not, why?

Node<Circle> nc = new Node<>();

Node<Shape> ns = nc;

Consider this class:

1. class Node<T> implements Comparable<T> {
2. public int compareTo(T obj) { /\* ... \*/ }
3. // ...
4. }

Will the following code compile? If not, why?

Node<String> node = new Node<>();

Comparable<String> comp = node;

How do you invoke the following method to find the first integer in a list that is relatively prime to a list of specified integers?

1. public static <T>
2. int findFirst(List<T> list, int begin, int end, UnaryPredicate<T> p)

Note that two integers *a* and *b* are relatively prime if gcd(*a, b*) = 1, where gcd is short for greatest common divisor.