Section A

Q1	12/14
Q2	20/20
Q3	10/10
Q4	6 /6

Compilation penalty
Style penalty (capped at -3)

Total for Section A	48 / 50

Really excellent just remember to use 2D Array etc when the situation arises

```
Username: jlk21

Compilation: 1 / 1

Model Answer’s Tests - Question1Tests: 6 / 6

Model Answer’s Tests - Question2Tests: 3 / 3

Model Answer’s Tests - Question3Tests: 3 / 3

Model Answer’s Tests - Question4Tests: 8 / 8

Model Answer’s Tests - extratestsformarking.MatrixIsEncapsulatedTest: 4 / 4

No Google style violations - excellent!


Style penalty (capped at -3): 0

Note: if it is below the cap, your total style penalty could be higher if the marker has
stylistic concerns that go beyond what Checkstyle identifies automatically.
```

```java
package generalmatrices;

import java.util.List;
import java.util.function.BinaryOperator;

public class ExampleMethods {

  public static Matrix<Matrix<Integer>> multiplyNestedMatrices(Matrix<Matrix<Integer>> first,
      Matrix<Matrix<Integer>> second) {
    BinaryOperator<Matrix<Integer>> intMatrixSum = (a, b) -> a.sum(b, Integer::sum);
    BinaryOperator<Matrix<Integer>> intMatrixProduct =
        (a, b) -> a.product(b, Integer::sum, (m, n) -> m * n);
    return first.product(second, intMatrixSum, intMatrixProduct);
  }

  public static Matrix<Pair> multiplyPairMatrices(List<Matrix<Pair>> matrices) {
    BinaryOperator<Pair> pairSum =
        (a, b) -> new Pair(a.getCoordX() + b.getCoordX(), a.getCoordY() + b.getCoordY());
    BinaryOperator<Pair> pairProduct =
        (a, b) -> new Pair(a.getCoordX() * b.getCoordX(), a.getCoordY() * b.getCoordY());
    return matrices.stream().reduce((a, b) -> a.product(b, pairSum, pairProduct)).get();
  }

}
```

✓ Perfect

10

✓

```java
package generalmatrices;

import java.util.ArrayList;
import java.util.List;
import java.util.function.BinaryOperator;

public final class Matrix<T> {
  private final List<T> elements;
  private final int order;

  public Matrix(List<T> elements) {
    if (elements.isEmpty()) {
      throw new IllegalArgumentException("List must be non-empty.");
    }
    if (!isPerfectSquare(elements.size())) {
      throw new IllegalArgumentException("List size must be a perfect square.");
    }
    this.elements = elements;
    order = (int) Math.sqrt(elements.size());
  }

  public T get(int row, int col) {
    int index = row * order + col;
    return elements.get(index);
  }

  public int getOrder() {
    return order;
  }

  public Matrix<T> sum(Matrix<T> other, BinaryOperator<T> elementSum) {
    List<T> sums = new ArrayList<>();
    for (int row = 0; row < order; row++) {
      for (int col = 0; col < order; col++) {
        T thisElement = get(row, col);
        T otherElement = other.get(row, col);
        sums.add(elementSum.apply(thisElement, otherElement));
      }
    }
    return new Matrix<>(sums);
  }

  public Matrix<T> product(
      Matrix<T> other, BinaryOperator<T> elementSum, BinaryOperator<T> elementProduct) {
    List<T> products = new ArrayList<>();
    for (int row = 0; row < order; row++) {
      for (int col = 0; col < order; col++) {
        List<T> individualProducts = new ArrayList<>();
        for (int offset = 0; offset < order; offset++) {
          T thisElement = get(row, offset);
          T otherElement = other.get(offset, col);
          individualProducts.add(elementProduct.apply(thisElement, otherElement));
        }
        T product = individualProducts.stream().reduce(elementSum::apply).get();
        products.add(product);
      }
    }
    return new Matrix<>(products);
  }

  @Override
  public String toString() {
    StringBuilder sb = new StringBuilder();
    sb.append("[");
    for (int row = 0; row < order; row++) {
      sb.append("[");
      for (int col = 0; col < order; col++) {
        sb.append(get(row, col).toString());
        if (col != order - 1) {
          sb.append(" ");
        }
      }
      sb.append("]");
    }
    sb.append("]");
    return sb.toString();
  }

  @Override
  public boolean equals(Object that) {
    if (!(that instanceof Matrix<?> thatMatrix)) {
```

→ Should use 2D Array or List

→ unnecessary if You used 2D Array or List

Could combine

12

20

✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓

```java
      return false;
    }
    if (order != thatMatrix.order) {
      return false;
    }
    for (int row = 0; row < order; row++) {
      for (int col = 0; col < order; col++) {
        T thisElement = get(row, col);
        if (!thisElement.equals(thatMatrix.get(row, col))) {
          return false;
        }
      }
    }
    return true;
  }

  @Override
  public int hashCode() {
    return Integer.hashCode(order) + elements.hashCode();
  }

  private boolean isPerfectSquare(int k) {
    double squareRoot = Math.sqrt(k);
    return Math.floor(squareRoot) == squareRoot;
  }
}
```

6

✓

```
Model Answer’s Tests - Question1Tests works!

JUnit version 4.12
......
Time: 0.018

OK (6 tests)


Model Answer’s Tests - Question2Tests works!

JUnit version 4.12
...
Time: 0.022

OK (3 tests)


Model Answer’s Tests - Question3Tests works!

JUnit version 4.12
...
Time: 0.021

OK (3 tests)


Model Answer’s Tests - Question4Tests works!

JUnit version 4.12
........
Time: 0.005

OK (8 tests)


Model Answer’s Tests - extratestsformarking.MatrixIsEncapsulatedTest works!

JUnit version 4.12
....
Time: 0.013

OK (4 tests)


```