OCP QUESTION 4

Given the code fragment:

```
List<Integer> numbers = Arrays.asList(1, 2, 3);
numbers.stream()
       .map(num -> num*2)                // line n1
       .peek(System.out::print)          // line n2
       .count();
```

What is the result?

A. 246
B. The code produces no output.
C. A compilation error occurs at line n1
D. A compilation error occurs at line n2

OCP QUESTION 7

Given the code fragments:

```
class OS {
    String name;
    OS (String name) {
        this.name = name;
    }
}
```

and

```
List<OS> list = Arrays.asList(
    new OS("Windows "),
    new OS("Solaris "),
    new OS("Linux ")
);
Stream<OS> creek = list.stream();
//line n1
```

Which should be inserted at line n1 to print Windows Solaris Linux ?

A. `creek.forEach(System.out::print);`
B. creek.map(a -> a.name).forEach(System.out::print);
C. `creek.map(a -> a).forEachOrdered(System.out::print);`
D. `creek.forEachOrdered(System.out::print);`

OCP QUESTION 49

Given the code fragment:

```
List<String> listVal = Arrays.asList("Joe", "Paul", "Alice", "Tom");
System.out.println (
     // line n1
);
```

Which code fragment, when inserted at line n1, enables the code to print the count of string elements whose length is greater than three?

A. listVal.stream().filter(x -> x.length()>3).count()
B. listVal.stream().map(x -> x.length()>3).count()
C. listVal.stream().peek(x -> x.length()>3).count().get()
D. listVal.stream().filter(x -> x.length()>3).mapToInt(x -> x).count()

OCP QUESTION 50

Which statement is true about `java.util.stream.Stream`?

A. A stream cannot be consumed more than once.
B. The execution mode of streams can be changed during processing.
C. Streams are intended to modify the source data.
D. A parallel stream is always faster than an equivalent sequential stream.

OCP QUESTION 52

Given the code fragment:

```
Path file = Paths.get("passwords.txt");
// line n1
```

Assume the passwords.txt is accessible. Which code fragment can be inserted at line n1 to enable the code to print the content of the passwords.txt file?

A. `List<String> fc = Files.list(file);`
`fc.stream().forEach(x -> System.out.println(x));`

B. `Stream<String> fc = Files.readAllLines(file);`
`fc.forEach(x -> System.out.println(x));`

C. `List<String> fc = readAllLines(file);`
`fc.stream().forEach(x -> System.out.println(x));`

D. `Stream<String> fc = Files.lines(file);`
`fc.forEach(x -> System.out.println(x));`

OCP QUESTION 73

Given the code fragment:

```
Stream<List<String>> listStream = Stream.of(
        Arrays.asList("1", "Smith"),
        Arrays.asList("2", null));
IntStream intStream = listStream.flatMapToInt((x) -> x.stream());
intStream.forEach(System.out::print);
```

What is the result?

A. 1Smith2null
B. 12
C. A NullPointerException is thrown at run time
D. A compilation error occurs

OCP QUESTION 76

Given the code fragment:

```
List<String> names = Arrays.asList("Alice", "Bob", "Chuck");
Function<String, String> func = x -> "Hi ".concat(x);
names.stream()
      .map(func)
      .peek(System.out::println);
```

What is the result?

A. Hi Alice
Hi Bob
Hi Chuck

B. Alice
Bob
Chuck

C. The program prints nothing

D. A compilation error occurs

OCP QUESTION 78

Given the code fragment:

```
List<String> planets = Arrays.asList("Mercury, 0",
        "Venus, 0",
        "Earth, 1",
        "Mars, 2");
planets.stream()
       .filter(x -> x.contains("M"))
       .sorted()
       .forEach(System.out::println);    //line n1
```

What is the result?

A. Mars, 2
Mercury, 0
B. A compilation error occurs at line n1
C. Mercury, 0
Venus, 0
Earth, 1
Mars, 2
D. Earth, 0
Venus, 0

OCP QUESTION 87

Given the code fragment:

```
List<Integer> list = Arrays.asList(13, 6, 62);
System.out.println(
        //line n1
);
```

Which code fragment must be inserted at line n1 to enable the code to print the minimum number in the list object?

A. `list.stream().min(Comparator.comparing(x -> x)).get()`
B. `list.stream().min(Integer::min).get()`
C. `list.stream().min()`
D. `list.stream().map(x -> x).min()`

OCP QUESTION 97

Given the code fragment:

```
List<String> archives = Arrays.asList("ZIP", "RAR", "TAR");
archives.forEach(x -> System.out.print(x + " "));
String common = archives.stream()
     .filter(x -> x.contains("AR"))
     .reduce((x, y) -> x + y).get();
System.out.println("\n" + common);
```

What is the result?

A. ZIP RAR TAR
RARTAR
B. ZIP RAR RARTAR
RARRARTAR
C. RARTAR
RARTAR
D. The order of the output is unpredictable.

OCP QUESTION 99

Given:

```
class Car {
    private List<Wheel> wheels;
    public Car(){ wheels = Arrays.asList(
        new Wheel(), new Wheel(), new Wheel(), new Wheel());
    }
    public List<Wheel> getWheels() {
        return wheels;
    }
}
class Wheel {
    private int airPressure;
    public Wheel(){
        airPressure = (int)(Math.random()*100);  // sets random values
    }                                            // from 0 to 99 incl.
    public int getAirPressure() {
        return airPressure;
    }
}
class Test{
    public static void main(String[] args) {
       List<Car> cars = Arrays.asList(new Car(), new Car(), new Car());
       System.out.println(cars.stream()
                              .map(Car::getWheels)              // line n1
                              .flatMap(Wheel::stream)           // line n2
                              .mapToInt(Wheel::getAirPressure) // line n3
                              .max()                            // line n4
                              .isPresent()                      // line n5
      );
   }
}
```

Which two modifications, when applied together, will let the code find the maximum value of air pressure of all wheels in all the cars?

A. Remove line n5

B. Replace line n5 with
`.orElse(12345)`

C. Replace line n2 with
`.flatMapToInt(List::stream)`

D. Replace line n2 with
`.flatMap(List::stream)`

E. Replace line n3 with
`.flatMap(Wheel::getAirPressure)`

F. Replace line n5 with
`.ifPresent(Wheel::getAirPressure)`

OCP QUESTION 100

Given in ForrestGump.java:

```
String str = "I been a idiot since I was born. My IQ is near 70, which
qualifies me, so they say.";                                 // line n1
Optional<String> result = Stream.of(
    str.split("[ ,.]")).anyMatch(x->x.startsWith("I"));      // line n2
System.out.println(result.get());                            // line n3
```

Which one of the following statements is correct?

A. The code prints either I or IQ.

B. The code always prints I.

C. The code prints I if lines n2 and n3 are changed to:

```
    String result = Stream.of(sentence.split("[ ,.]"))
                          .anyMatch(x->x.startsWith("I"));
    System.out.println(result.get());
```

D. The code prints either I or IQ if lines n2 and n3 are changed to:

```
    Optional<String> result = Stream.of(str.split("[ ,.]"))
                           .parallel()
                           .anyMatch(x->x.startsWith("I"));
    System.out.println(result.get());
```

E. The code fails to compile.