Java Stream API - 0

Given code of Test.java file:

```
import java.util.stream.Stream;

public class Test {
    public static void main(String[] args) {
        Stream<Integer> stream = Stream.iterate(1, i -> i + 1);
        System.out.println(stream.anyMatch(i -> i > 1));
    }
}
```

What will be the result of compiling and executing Test class?

A - true is printed on to the console and code runs infinitely

B - true

C - Nothing is printed on to the console as code runs infinitely

D - false

Java Stream API - 1

Given code of Test.java file:

```
import java.util.Arrays;
import java.util.List;
import java.util.function.Predicate;

public class Test {
    public static void main(String[] args) {
        List<Integer> list = Arrays.asList(-80, 100, -40, 25, 200);
        Predicate<Integer> predicate = num -> {
            int ctr = 1;
            boolean result = num > 0;
            System.out.print(ctr++ + ".");
            return result;
        };

        list.stream().filter(predicate).findFirst();
    }
}
```

What will be the result of compiling and executing Test class?

A - 1.

B - 1.1.1.1.1.

C - 1.1.

D - Nothing is printed on to the console.

E - 1.2.

F - 1.2.3.4.5.

Java Stream API - 2

Given code of Test.java file:

```
import java.util.OptionalDouble;

class MyException extends RuntimeException{}

public class Test {
    public static void main(String[] args) {
        OptionalDouble optional = OptionalDouble.empty();
        System.out.println(optional.orElseThrow(MyException::new));
    }
}
```

What will be the result of compiling and executing Test class?

A - Compilation error

B - An instance of NoSuchElementException is thrown at runtime

C - An instance of MyException is thrown at runtime

D - An instance of NullPointerException is thrown at runtime

E - An instance of RuntimeException is thrown at runtime

F - null

Java Stream API - 3

Given code of Test.java file:

```
import java.util.stream.IntStream;

public class Test {
    public static void main(String[] args) {
        int res = 1;
        IntStream stream = IntStream.rangeClosed(1, 5);
        /*INSERT*/
    }
}
```

Which of the following options can replace /*INSERT*/ such that on executing Test class, 120 is printed in the output? NOTE: 120 is the multiplication of numbers from 1 to 5. Select 2 options.

A - System.out.println(stream.reduce(1, Integer::multiply));

B - System.out.println(stream.reduce(1, (i, j) -> i* j));

C - System.out.println(stream.reduce(res, (i, j) -> i * j));

D - System.out.println(stream.reduce(0, Integer::multiply));

E - System.out.println(stream.reduce(0, (i, j) -> i* j));

Java Stream API - 4

Given code of Test.java file:

```
import java.util.Set;
import java.util.stream.Collectors;
import java.util.stream.Stream;

public class Test {
    public static void main(String[] args) {
        Stream<String> stream = Stream.of("java", "python", "c",
                "c++", "java", "python");
        Set<String> set = stream.collect(Collectors.toSet());
        System.out.println(set.size());
    }
}
```

What will be the result of compiling and executing Test class?

A - 0

B - 4

C - 5

D - 6

Java Stream API - 5

Given code of Test.java file:

```
import java.util.ArrayList;
import java.util.List;

class Point {
    int x;
    int y;
    Point(int x, int y) {
        this.x = x;
        this.y = y;
    }

    public String toString() {
        return "Point(" + x + ", " + y + ")";
    }

    boolean filter() {
        return this.x == this.y;
    }

}
public class Test {
    public static void main(String[] args) {
        List<Point> list = new ArrayList<>();
        list.add(new Point(0, 0));
        list.add(new Point(1, 2));
        list.add(new Point(-1, -1));


         list.stream().filter(Point::filter).forEach(System.out::println);
          //Line n1
    }
}
```

What will be the result of compiling and executing Test class?

A -

```
Point(0, 0)
Point(-1, -1)
```

B -

```
Point(0, 0)
Point(1, 2)
Point(-1, -1)
```

C - Line n1 causes compilation error

D - Point(1, 2)

Java Stream API - 6

Given code of Test.java file:

```
import java.util.ArrayList;
import java.util.List;

public class Test {
    public static void main(String[] args) {
        List<String> list = new ArrayList<>();
        System.out.println(list.stream().anyMatch(s -> s.length() >
        0));
        System.out.println(list.stream().allMatch(s -> s.length() >
        0));
        System.out.println(list.stream().noneMatch(s -> s.length() >
        0));
    }
}
```

What will be the result of compiling and executing Test class?

A -

```
false
true
true
```

B -

```
true
true
true
```

C -

```
true
false
false
```

D -

```
false
false
false
```

Java Stream API - 7

Given code of Test.java file:

```
import java.util.stream.IntStream;

public class Test {
    public static void main(String[] args) {
        int sum = IntStream.rangeClosed(1,3).map(i -> i * i)
                .map(i -> i * i).sum();
        System.out.println(sum);
    }
}
```

What will be the result of compiling and executing Test class?

A - 98

B - 6

C - None of the other options

D - 14

Java Stream API - 8

Given code of Test.java file:

```
import java.util.Arrays;
import java.util.List;
import java.util.stream.Collectors;

class Book {
    String title;
    String author;
    double price;

    public Book(String title, String author, double price) {
        this.title = title;
        this.author = author;
        this.price = price;
    }

    public String getAuthor() {
        return this.author;
    }

    public String toString() {
        return "{" + title + "," + author + "," + price + "}";
    }
}

public class Test {
    public static void main(String[] args) {
        List<Book> books = Arrays.asList(
                new Book ("Head First Java", "Kathy Sierra", 24.5),
                new Book ("OCP", "Udayan Khattry", 20.99),
                new Book ("OCA", "Udayan Khattry", 14.99));

         books.stream().collect(Collectors.groupingBy(Book::getAuthor)).forEach(
          -> System.out.println(a));
    }
}
```

What will be the result of compiling and executing Test class?

A - Runtime Exception

B -

```
Kathy Sierra
Udayan Khattry
```

C -

```
[{Head First Java,Kathy Sierra,24.5}]
[{OCP,Udayan Khattry,20.99}, {OCA,Udayan Khattry,14.99}]
```

Java Stream API - 9

Given code of Test.java file:

```
import java.util.Optional;

public class Test {
    public static void main(String[] args) {
        Optional<Integer> optional = Optional.ofNullable(null);
        System.out.println(optional);
    }
}
```

What will be the result of compiling and executing Test class?

A - Optional[0]

B - Optional.empty

C - NullPointerException is thrown at runtime

D - Optional[null]

Java Stream API - 10

Given code of Test.java file:

```
import java.util.HashMap;
import java.util.Map;

public class Test {
    public static void main(String[] args) {
        Map<Integer, String> map = new HashMap<>();
        map.put(1, "ONE");
        map.put(2, "TWO");
        map.put(3, "THREE");

        System.out.println(map.stream().count());
    }
}
```

What will be the result of compiling and executing Test class?

A - 3

B - Runtime Exception

C - Compilation error

D - 6

Java Stream API - 11

Given code of Test.java file:

```
import java.util.Arrays;
import java.util.List;

public class Test {
    public static void main(String[] args) {
        List<String> codes = Arrays.asList("1st", "2nd", "3rd",
        "4th");
        System.out.println(codes.stream().filter(
                s -> s.endsWith("d")).reduce((s1, s2) -> s1 + s2));
    }
}
```

What will be the result of compiling and executing Test class?

A - Optional[1st4th]

B - 1st4th

C - 2nd3rd

D - Optional[2nd3rd]

Java Stream API - 12

Given code of Test.java file:

```
import java.util.ArrayList;
import java.util.List;

class Rope {
    int length;
    String color;

    Rope(int length, String color) {
        this.length = length;
        this.color = color;
    }

    public String toString() {
        return "Rope [" + color + ", " + length + "]";
    }

    static class RedRopeFilter {
        boolean filter(Rope rope) {
            return rope.color.equalsIgnoreCase("Red");
        }
    }
}

public class Test {
    public static void main(String[] args) {
        List<Rope> list = new ArrayList<>();
        list.add(new Rope(5, "red"));
        list.add(new Rope(10, "Red"));
        list.add(new Rope(7, "RED"));
        list.add(new Rope(10, "green"));
        list.add(new Rope(7, "Blue"));

        list.stream().filter(new
         Rope.RedRopeFilter()::filter).forEach(System.out::println);
        //Line n1
    }
}
```

What will be the result of compiling and executing Test class?

A -

```
Rope [red, 5]
Rope [Red, 10]
Rope [RED, 7]
```

B -

```
Rope [green, 10]
```

```
Rope [Blue, 7]
```

C -

```
Rope [red, 5]
Rope [Red, 10]
Rope [RED, 7]
Rope [green, 10]
Rope [Blue, 7]
```

D -

```
Rope [Red, 10]
```

E - Line n1 causes compilation error

Java Stream API - 13

Given code of Test.java file:

```
import java.util.Arrays;
import java.util.stream.Stream;

public class Test {
    public static void main(String[] args) {
        Stream<Integer> stream = Arrays.asList(1,2,3,4,5).stream();
        System.out.println(stream.sum());
    }
}
```

What will be the result of compiling and executing Test class?

A - Runtime Exception

B - 15

C - Compilation error

Java Stream API - 14

Given code of Test.java file:

```
import java.util.Arrays;
import java.util.List;
import java.util.Map;
import java.util.stream.Collectors;

enum Color {
    RED, YELLOW, GREEN
}

class TrafficLight {
    String msg;
    Color color;

    TrafficLight(String msg, Color color) {
        this.msg = msg;
        this.color = color;
    }

    public String getMsg() {
        return msg;
    }

    public Color getColor() {
        return color;
    }

    public String toString() {
        return "{" + color + ", " + msg + "}";
    }
}

public class Test {
    public static void main(String[] args) {
        TrafficLight tl1 = new TrafficLight("Go", Color.GREEN);
        TrafficLight tl2 = new TrafficLight("Go Now!", Color.GREEN);
        TrafficLight tl3 = new TrafficLight("Ready to stop",
        Color.YELLOW);
        TrafficLight tl4 = new TrafficLight("Slow Down",
        Color.YELLOW);
        TrafficLight tl5 = new TrafficLight("Stop", Color.RED);

        List<TrafficLight> list = Arrays.asList(tl1, tl2, tl3, tl4,
        tl5);

        Map<Color, List<String>> map = list.stream()

        .collect(Collectors.groupingBy(TrafficLight::getColor,
                      Collectors.mapping(TrafficLight::getMsg,
        Collectors.toList())));
```

```
        System.out.println(map.get(Color.YELLOW));
    }
}
```

What will be the result of compiling and executing Test class?

A - [Slow Down]

B - Some text containing @ symbol

C - [Ready to stop, Slow Down]

D - [Ready to stop]

Java Stream API - 15

Given code of Test.java file:

```
import java.util.OptionalInt;

class MyException extends Exception{}

public class Test {
    public static void main(String[] args) {
        OptionalInt optional = OptionalInt.empty();
        System.out.println(optional.orElseThrow(MyException::new));
    }
}
```

What will be the result of compiling and executing Test class?

A - An instance of MyException is thrown at runtime

B - An instance of RuntimeException is thrown at runtime

C - Compilation error

D - null

E - An instance of NoSuchElementException is thrown at runtime

F - An instance of NullPointerException is thrown at runtime

Java Stream API - 16

Given code of Test.java file:

```
import java.util.stream.IntStream;

public class Test {
    public static void main(String[] args) {
        int res = 1;
        IntStream stream = IntStream.rangeClosed(1, 4);

        System.out.println(stream.reduce(res++, (i, j) -> i * j));
    }
}
```

What will be the result of compiling and executing Test class?

A - 6

B - Compilation error as res should be effectively final

C - 48

D - 12

E - 24

Java Stream API - 17

Given code of Test.java file:

```
import java.util.Arrays;
import java.util.List;
import java.util.function.Predicate;

public class Test {
    public static void main(String[] args) {
        List<Integer> list = Arrays.asList(-80, 100, -40, 25, 200);
        Predicate<Integer> predicate = num -> {
            int ctr = 1;
            boolean result = num > 0;
            System.out.print(ctr++ + ".");
            return result;
        };

        list.stream().filter(predicate).sorted();
    }
}
```

What will be the result of compiling and executing Test class?

A - 1.2.3.4.5.

B - 1.1.

C - 1.

D - 1.2.

E - Nothing is printed on to the console

F - 1.1.1.1.1.

Java Stream API - 18

Given code of Test.java file:

```
import java.util.List;
import java.util.Map;
import java.util.stream.Collectors;
import java.util.stream.Stream;

class Certification {
    String studId;
    String test;
    int marks;

    Certification(String studId, String test, int marks) {
        this.studId = studId;
        this.test = test;
        this.marks = marks;
    }

    public String toString() {
        return "{" + studId + ", " + test + ", " + marks + "}";
    }

    public String getStudId() {
        return studId;
    }

    public String getTest() {
        return test;
    }

    public int getMarks() {
        return marks;
    }
}

public class Test {
    public static void main(String[] args) {
        Certification c1 = new Certification("S001", "OCA", 87);
        Certification c2 = new Certification("S002", "OCA", 82);
        Certification c3 = new Certification("S001", "OCP", 79);
        Certification c4 = new Certification("S002", "OCP", 89);
        Certification c5 = new Certification("S003", "OCA", 60);
        Certification c6 = new Certification("S004", "OCA", 88);

        Stream<Certification> stream = Stream.of(c1, c2, c3, c4, c5,
         c6);
        Map<Boolean, List<Certification>> map =
                stream.collect(Collectors.partitioningBy(s ->
         s.equals("OCA")));
        System.out.println(map.get(true));
    }
```

```
}
```

What will be the result of compiling and executing Test class?

A - [{S001, OCP, 79}, {S002, OCP, 89}]

B - [{S001, OCA, 87}, {S002, OCA, 82}, {S001, OCP, 79}, {S002, OCP, 89}, {S003, OCA, 60}, {S004, OCA, 88}]

C - []

D - [{S001, OCA, 87}, {S002, OCA, 82}, {S003, OCA, 60}, {S004, OCA, 88}]

Java Stream API - 19

Given code of Test.java file:

```
import java.util.ArrayList;
import java.util.List;

class MyString {
    String str;
    MyString(String str) {
        this.str = str;
    }
}

public class Test {
    public static void main(String[] args) {
        List<MyString> list = new ArrayList<>();
        list.add(new MyString("Y"));
        list.add(new MyString("E"));
        list.add(new MyString("S"));

        list.stream().map(s -> s).forEach(System.out::print);
    }
}
```

Which of the following statements are correct?

A - Above code causes compilation error

B - Above code terminates successfully without printing anything on to the console

C - On execution, above code prints “YES” on to the console

D - Above code terminates successfully after printing text other than “YES” on to the console

Java Stream API - 20

Given code of Test.java file:

```
import java.util.Optional;
import java.util.stream.Stream;

public class Test {
    public static void main(String[] args) {
        Stream<Number> stream = Stream.of();
        Optional<Number> optional = stream.findFirst();
        System.out.println(optional.orElse(-1));
    }
}
```

What will be the result of compiling and executing Test class?

A - null

B - Optional.empty

C - 0

D - -1

Java Stream API - 21

Given code of Test.java file:

```
import java.util.Arrays;
import java.util.Comparator;
import java.util.List;

public class Test {
    public static void main(String[] args) {
        List<Integer> list = Arrays.asList(10, 20, 8);

        System.out.println(list.stream().max(Comparator.comparing(a
        -> a)).get()); //Line 1


         System.out.println(list.stream().max(Integer::compareTo).get());
          //Line 2

        System.out.println(list.stream().max(Integer::max).get());
         //Line 3
    }
}
```

Which of the following statement is true?

A - Line 1 and Line 3 print same output

B - Line 1 and Line 2 print same output

C - Line 2 and Line 3 print same output

D - Line 1, Line 2 and Line 3 print same output

Java Stream API - 22

Given code of Test.java file:

```
import java.util.stream.Stream;

public class Test {
    public static void main(String[] args) {
        System.out.println(Stream.of(10, 0,
         -10).sorted().findAny().orElse(-1));
    }
}
```

Which of the following statements are true about the execution of Test class?

A - It will always print 0 on to the console.

B - It can print any number from the stream.

C - It will always print 10 on to the console.

D - It will never print -1 on to the console.

E - It will always print -10 on to the console.

Java Stream API - 23

Given code of Test.java file:

```
import java.util.ArrayList;
import java.util.List;

public class Test {
    public static void main(String[] args) {
        int ref = 10;
        List<Integer> list = new ArrayList<>();
        list.stream().anyMatch(i -> {
            System.out.println("HELLO");
            return i > ref;
        });
    }
}
```

What will be the result of compiling and executing Test class?

A - Program executes successfully but nothing is printed on to the console

B - HELLO

C - Compilation error

Java Stream API - 24

Given code of Test.java file:

```
import java.util.Optional;

public class Test {
    public static void main(String[] args) {
        Optional<Integer> optional = Optional.of(null); //Line 8
        System.out.println(optional.orElse(-1)); //Line 9
    }
}
```

What will be the result of compiling and executing Test class?

A - Line 9 throws NullPointerException

B - null

C - -1

D - Line 8 throws NullPointerException

Java Stream API - 25

Which of the following are Primitive variant of Optional class?

A - ByteOptional

B - OptionalBoolean

C - OptionalDouble

D - IntOptional

E - OptionalFloat

Java Stream API - 26

Given code of Test.java file:

```
import java.util.Set;
import java.util.TreeSet;
import java.util.stream.Collectors;
import java.util.stream.Stream;

public class Test {
    public static void main(String[] args) {
        Stream<String> stream = Stream.of("java", "python", "c",
                "c++", "java", "python");
        Set<String> set =
         stream.collect(Collectors.toCollection(TreeSet::new));
        System.out.println(set);
    }
}
```

What will be the result of compiling and executing Test class?

A - [java, python, c, c++]

B - [c, c++, java, python]

C - [c++, c, java, python]

D - Order of elements can’t be predicted in the output.

Java Stream API - 27

Given code of Test.java file:

```
import java.util.stream.Stream;

public class Test {
    public static void main(String[] args) {
        Stream<String> stream = Stream.of("ocp");
        stream._____________(s -> s.chars()).forEach(i ->
         System.out.print((char)i));
    }
}
```

Which code snippet, when filled into the blank, allows the class to compile?

A - flatMap

B - flatMapToDouble

C - flatMapToLong

D - flatMapToInt

Java Stream API - 28

Given code of Test.java file:

```
import java.util.ArrayList;
import java.util.Comparator;
import java.util.List;

class Fruit implements Comparable<Fruit>, Comparator<Fruit> {
    String name;
    String countryOfOrigin;

    Fruit() {}

    Fruit(String name, String countryOfOrigin) {
        this.name = name;
        this.countryOfOrigin = countryOfOrigin;
    }

    public String toString() {
        return name + ":" + countryOfOrigin;
    }

    @Override
    public int compareTo(Fruit o) {
        return this.name.compareToIgnoreCase(o.name);
    }

    @Override
    public int compare(Fruit o1, Fruit o2) {
        return
         o1.countryOfOrigin.compareToIgnoreCase(o2.countryOfOrigin);
    }

    public static int comp(String s1, String s2) {
        return s2.compareToIgnoreCase(s1);
    }
}

public class Test {
    public static void main(String[] args) {
        List<Fruit> list = new ArrayList<>();
        list.add(new Fruit("Olive", "Middle East"));
        list.add(new Fruit("Mango", "India"));
        list.add(new Fruit("Cranberry", "North America"));
        list.add(new Fruit("Watermelon", "Africa"));
        list.add(new Fruit("Peach", "China"));
        list.add(new Fruit("Fig", "Middle East"));
        list.add(new Fruit("Blueberry", "North America"));

        /* INSERT */
    }
}
```

Which of the following two options can replace /* INSERT */ such that output is:

```
Cranberry:North America
Blueberry:North America
Olive:Middle East
Fig:Middle East
Mango:India
Peach:China
Watermelon:Africa
```

A - list.stream().sorted(Comparator.comparing(f -> f.countryOfOrigin, Fruit::comp)).forEach(System.out::println);

B - list.stream().sorted(new Fruit().reversed()).forEach(System.out::println);

C - list.stream().sorted(new Fruit()).forEach(System.out::println);

D - list.stream().sorted().forEach(System.out::println);

Java Stream API - 29

Given code of Test.java file:

```
import java.util.stream.Stream;

public class Test {
    public static void main(String[] args) {
        Stream<String> stream = Stream.of("d", "cc", "bbb", "aaaa");
        stream.sorted().forEach(System.out::println);
    }
}
```

Which of the following needs to be done, so that output is:

```
d
cc
bbb
aaaa
```

A - Replace stream.sorted() with stream.sorted((s1,s2) -> s1.length() - s2.length())

B - No need to make any changes, on execution given code prints expected result

C - Replace stream.sorted() with stream.sorted((s1,s2) -> s2.length() - s1.length())

Java Stream API - 30

Given code of Test.java file:

```
import java.util.Optional;

class Message {
    private String msg = "Good Morning!";
    public Message(String msg) {
        this.msg = msg;
    }

    public Message() {super();}

    public String getMsg() {
        return msg;
    }

    public String toString() {
        return msg;
    }
}

public class Test {
    public static void main(String[] args) {
        Message message = null;
        Optional<Message> optional = Optional.ofNullable(message);
        System.out.println(optional.isPresent() ?
         optional.get().getMsg() : new Message());
    }
}
```

What will be the result of compiling and executing Test class?

A - null

B - Text containing @ symbol

C - NullPointerException is thrown at runtime

D - Good Morning!

Java Stream API - 31

Given code of Test.java file:

```
import java.util.ArrayList;
import java.util.List;

public class Test {
    public static void main(String[] args) {
        int ref = 10;
        List<Integer> list = new ArrayList<>();
        list.stream().anyMatch(i -> {
            System.out.println("HELLO");
            return i > ++ref;
        });
    }
}
```

What will be the result of compiling and executing Test class?

A - HELLO

B - Compilation error

C - Program executes successfully but nothing is printed on to the console

Java Stream API - 32

Given code of Test.java file:

```
import java.util.Arrays;
import java.util.List;
import java.util.function.Predicate;

public class Test {
    public static void main(String[] args) {
        List<Integer> list = Arrays.asList(-80, 100, -40, 25, 200);
        Predicate<Integer> predicate = num -> {
            int ctr = 1;
            boolean result = num > 0;
            System.out.print(ctr++ + ".");
            return result;
        };

        list.stream().filter(predicate).count();
    }
}
```

What will be the result of compiling and executing Test class?

A - 1.2.3.

B - 2.4.5.

C - 1.1.1.

D - 1.1.1.1.1.

E - 1.2.3.4.5.

Java Stream API - 33

Given code of Test.java file:

```
import java.util.Arrays;
import java.util.stream.Stream;

public class Test {
    public static void main(String[] args) {
        Stream<Integer> stream = Arrays.asList(1,2,3,4,5).stream();
        System.out.println(stream.mapToInt(i ->
         i).average().getAsInt());
    }
}
```

What will be the result of compiling and executing Test class?

A - 3

B - Compilation error

C - Runtime Exception

Java Stream API - 34

Given code of Test.java file:

```
import java.util.Arrays;
import java.util.stream.Stream;

public class Test {
    public static void main(String[] args) {
        Stream<String> stream = Arrays.asList("One", "Two",
        "Three").stream();
        System.out.println(stream.reduce(null, (s1, s2) -> s1 +
        s2));
    }
}
```

What will be the result of compiling and executing Test class?

A - NullPointerException is thrown at runtime

B - OneTwoThreenull

C - nullOneTwoThree

D - OneTwoThree

Java Stream API - 35

Given code of Test.java file:

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

public class Test {
    public static void main(String[] args) {
        List<Boolean> list = Arrays.asList(false, new Boolean(null),
         new Boolean("1"), new Boolean("0"));
        BinaryOperator<Boolean> operator = (b1, b2) -> b1 || b2;
        System.out.println(list.stream().reduce(false, operator));
    }
}
```

What will be the result of compiling and executing Test class?

A - NullPointerException is thrown at runtime

B - Compilation error

C - true

D - false

Java Stream API - 36

Given code of Test.java file:

```
import java.util.Optional;
import java.util.function.Predicate;
import java.util.stream.Stream;

public class Test {
    public static void main(String[] args) {
        Stream<String> stream = Stream.of("and", "Or", "not",
        "Equals", "unary", "binary");
        Optional<String> optional = stream.filter(

        ((Predicate<String>)Test::isFirstCharVowel).negate()).findFirst();
        System.out.println(optional.get());
    }

    private static boolean isFirstCharVowel(String str) {
        str = str.substring(0, 1).toUpperCase();
        switch(str) {
            case "A":
            case "E":
            case "I":
            case "O":
            case "U":
                return true;
        }
        return false;
    }
}
```

What will be the result of compiling and executing Test class?

A - not

B - Or

C - and

D - binary

Java Stream API - 37

Given code of Test.java file:

```
import java.util.List;
import java.util.stream.Collectors;
import java.util.stream.Stream;

public class Test {
    public static void main(String[] args) {
        Stream<String> stream = Stream.of("java", "python", "c",
        "c++");
        List<String> list =
         stream.sorted().collect(Collectors.toList());
        System.out.println(list);
    }
}
```

What will be the result of compiling and executing Test class?

A - [c, c++, java, python]

B - [java, python, c, c++]

C - [c++, c, java, python]

D - [python, java, c++, c]

Java Stream API - 38

Given code of Test.java file:

```
import java.util.Map;
import java.util.TreeMap;
import java.util.function.Function;
import java.util.stream.Collectors;
import java.util.stream.Stream;

class Person {
    int id;
    String name;
    Person(int id, String name) {
        this.id = id;
        this.name = name;
    }
    public String toString() {
        return "{" + id + ", " + name + "}";
    }

    public boolean equals(Object obj) {
        if(obj instanceof Person) {
            Person p = (Person) obj;
            return this.id == p.id;
        }
        return false;
    }

    public int hashCode() {
        return new Integer(this.id).hashCode();
    }
}

public class Test {
    public static void main(String[] args) {
        Person p1 = new Person(1010, "Sean");
        Person p2 = new Person(2843, "Rob");
        Person p3 = new Person(1111, "Lucy");

        Stream<Person> stream = Stream.of(p1, p2, p3);
        Map<Integer, Person> map = stream.collect(/*INSERT*/);
        System.out.println(map.size());
    }
}
```

Which of the following statements can replace /*INSERT*/ such that output is 3? 1. Collectors.toMap(p -> p.id, Function.identity()) 2. Collectors.toMap(p -> p.id, p -> p) 3. Collectors.toCollection(TreeMap::new)

A - Only 3

B - Both 1 & 2

C - Both 2 & 3

D - Only 1

E - Only 2

F - All 1, 2 & 3

Java Stream API - 39

Given code of Test.java file:

```
import java.util.stream.Stream;

public class Test {
    public static void main(String[] args) {
        Stream<StringBuilder> stream = Stream.of();
        stream.map(s -> s.reverse()).forEach(System.out::println);
    }
}
```

What will be the result of compiling and executing Test class?

A - NullPointerException is thrown at runtime

B - Compilation error

C - ClassCastException is thrown at runtime

D - Program executes successfully but nothing is printed on to the console

Java Stream API - 40

Given code of Test.java file:

```
import java.util.Arrays;
import java.util.stream.Stream;

public class Test {
    public static void main(String[] args) {
        Stream<Double> stream = Arrays.asList(1.8, 2.2,
         3.5).stream();

        /*INSERT*/
    }
}
```

Which of the following options can replace /*INSERT*/ such that on executing Test class, all the stream elements are added and result is printed on to the console?

Select ALL that apply.

A - System.out.println(stream.reduce(0.0, (d1, d2) -> d1 + d2));

B - System.out.println(stream.sum());

C - System.out.println(stream.reduce(0, (di, d2) -> di + d2));

D - System.out.println(stream.reduce(9, Double::sum));

E - System.out.println(stream.reduce(0.0, Double::sum));

Java Stream API - 41

Given code of Test.java file:

```
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

public class Test {
    public static void main(String[] args) {
        List<String> list = new ArrayList<>(Arrays.asList("Z", "Y",
        "X"));
        list.stream().sorted().findFirst().get();
        System.out.println(list.get(2));
    }
}
```

What will be the result of compiling and executing Test class?

A - Z

B - Runtime Exception

C - X

D - Y

Java Stream API - 42

Given code of Test.java file:

```
import java.util.Random;
import java.util.stream.IntStream;

public class Test {
    public static void main(String[] args) {
        IntStream stream = IntStream.generate(() -> new
        Random().nextInt(100))

        .limit(5);
        stream.filter(i -> i > 0 && i <
        10).findFirst()._________________;
    }
}
```

Which code snippet, when filled into the blank, allows the class to compile?

A - get()

B - ifPresent(System.out::println)

C - map(i -> i * i)

D - forEach(System.out::println)

Java Stream API - 43

Given code of Test.java file:

```
import java.util.Comparator;
import java.util.stream.Stream;

public class Test {
    public static void main(String[] args) {
        Stream<String> stream = Stream.of("d", "a", "mm", "bb",
        "zzz", "www");
        Comparator<String> lengthComp = (s1, s2) -> s1.length() -
         s2.length();
        stream.sorted(lengthComp).forEach(System.out::println);
    }
}
```

Which of the following needs to be done, so that output is:

```
a
d
bb
mm
www
zzz
```

A - No need to make any changes, on execution given code prints expected result.

B - Replace stream.sorted(lengthComp) with stream.sorted(lengthComp.thenComparing(String::compareTo))

C - Replace stream.sorted(lengthComp) with stream. sorted(lengthComp.reversed())

Java Stream API - 44

Given code of Test.java file:

```
import java.util.stream.IntStream;

public class Test {
    public static void main(String[] args) {
        IntStream stream = IntStream.rangeClosed(1, 20).filter(i ->
         i % 2 == 0);
        System.out.println(stream.summaryStatistics());
    }
}
```

Which of the following statements is true for above code?

A - On execution only sum and average data will be printed on to the console

B - On execution a text containing @ symbol will be printed on to the console

C - On execution only max, min and count data will be printed on to the console

D - On execution sum, average, max, min and count data will be printed on to the console

Java Stream API - 45

Given code of Test.java file:

```
import java.util.Optional;
import java.util.stream.Stream;

public class Test {
    public static void main(String[] args) {
        Optional<String> optional = Stream.of("red", "green",
        "blue", "yellow")
                .sorted().findFirst();
        System.out.println(optional.get());
    }
}
```

What will be the result of compiling and executing Test class?

A - red

B - green

C - yellow

D - blue

Java Stream API - 46

Given code of Test.java file:

```
import java.util.stream.Stream;

public class Test {
    public static void main(String[] args) {
        Stream<Double> stream = Stream.generate(() -> new
        Double("1.0")).limit(10);
        System.out.println(stream.filter(d -> d > 2).allMatch(d -> d
        == 2));
    }
}
```

What will be the result of compiling and executing Test class?

A - false

B - true

Java Stream API - 47

Given code of Test.java file:

```
import java.util.stream.Stream;

public class Test {
    public static void main(String[] args) {
        Stream.of().map(s ->
         s.reverse()).forEach(System.out::println);
    }
}
```

What will be the result of compiling and executing Test class?

A - Program executes successfully but nothing is printed on to the console

B - Compilation error

C - ClassCastException is thrown at runtime

D - NullPointerException is thrown at runtime

Java Stream API - 48

Given code of Test.java file:

```
import java.util.Optional;
import java.util.stream.Stream;

public class Test {
    public static void main(String[] args) {
        Optional<Integer> optional = Stream.of(10).findFirst();
        System.out.println(optional);
    }
}
```

What will be the result of compiling and executing Test class?

A - Optional[10]

B - Text containing @ symbol

C - 10

Java Stream API - 49

Given code of Test.java file:

```
import java.util.Comparator;
import java.util.stream.Stream;

public class Test {
    public static void main(String[] args) {
        Comparator<Integer> comp = (i1, i2) -> i2.compareTo(i1);
        Stream<Integer> stream = Stream.of(55, 23, -9, 8, 42);
        stream.sorted(comp.reversed()).forEach(i ->
         System.out.print(i + " "));
    }
}
```

What will be the result of compiling and executing Test class?

A - 42 8 -9 23 55

B - 55 42 23 8 -9

C - 55 23 -9 8 42

D - -9 8 23 42 55

Java Stream API - 50

Given code of Test.java file:

```
import java.util.List;
import java.util.Map;
import java.util.stream.Collectors;
import java.util.stream.Stream;

class Certification {
    String studId;
    String test;
    int marks;

    Certification(String studId, String test, int marks) {
        this.studId = studId;
        this.test = test;
        this.marks = marks;
    }

    public String toString() {
        return "{" + studId + ", " + test + ", " + marks + "}";
    }

    public String getStudId() {
        return studId;
    }

    public String getTest() {
        return test;
    }

    public int getMarks() {
        return marks;
    }
}

public class Test {
    public static void main(String[] args) {
        Certification c1 = new Certification("S001", "OCA", 87);
        Certification c2 = new Certification("S002", "OCA", 82);
        Certification c3 = new Certification("S001", "OCP", 79);
        Certification c4 = new Certification("S002", "OCP", 89);
        Certification c5 = new Certification("S003", "OCA", 60);
        Certification c6 = new Certification("S004", "OCA", 88);

        Stream<Certification> stream = Stream.of(c1, c2, c3, c4, c5,
         c6);
        Map<String, List<Certification>> map =

         stream.collect(Collectors.groupingBy(Certification::getTest));
        System.out.println(map.get("OCP"));
    }
```

```
}
```

What will be the result of compiling and executing Test class?

A - [{S001, OCP, 79}, {S002, OCP, 89}]

B - [{S001, OCA, 87}, {S002, OCA, 82}, {S003, OCA, 60}, {S004, OCA, 88}]

C - [{S001, OCA, 87}, {S002, OCA, 82}, {S001, OCP, 79}, {S002, OCP, 89}, {S003, OCA, 60}, {S004, OCA, 88}]

D - []

Java Stream API - 51

Given code of Test.java file:

```
import java.util.function.LongFunction;
import java.util.function.LongUnaryOperator;
import java.util.stream.Stream;

public class Test {
    public static void main(String[] args) {
        long seed = 10;
        Stream<Long> stream = Stream.iterate(seed, i -> i +
        2).limit(2); //Line n1
        LongFunction<LongUnaryOperator> func = m -> n -> n / m;
        //Line n2
        stream.mapToLong(i ->
        i).map(func.apply(2)).forEach(System.out::println); //Line
        n3
    }
}
```

What will be the result of compiling and executing Test class?

A -

```
1
1
```

B -

```
5
6
```

C -

```
10
12
```

D - Compilation error

E -

```
0
0
```

Java Stream API - 52

Given code of Test.java file:

```
import java.util.Arrays;
import java.util.IntSummaryStatistics;
import java.util.stream.Stream;

public class Test {
    public static void main(String[] args) {
        String text = "I am going to pass OCP exam in first attempt";
        Stream<String> stream = Arrays.stream(text.split(" "));
        IntSummaryStatistics stat = stream.map(s ->
         s.length()).summaryStatistics();
        System.out.println(stat.getMax());
    }
}
```

Which of the following needs to be done, so that output is 7?

A - Replace stream.map(s -> s.length()) with stream.mapToInt(s -> s.length())

B - No need to make any changes, on execution given code prints 7 on to the console

C - Replace text.split(" ") with text.split(",")

D - Replace stat.getMax() with stat.getCount()

Java Stream API - 53

Given code of Test.java file:

```
import java.util.stream.Stream;

public class Test {
    public static void main(String[] args) {
        Stream.of(true, false, true).map(b -> b.toString()
                .toUpperCase()).peek(System.out::println).count();
    }
}
```

What will be the result of compiling and executing Test class?

A -

```
TRUE
FALSE
TRUE
```

B -

```
true
false
true
3
```

C -

```
true
false
true
```

D -

```
TRUE
FALSE
TRUE
3
```

Java Stream API - 54

Given code of Test.java file:

```
import java.util.Random;
import java.util.stream.IntStream;

public class Test {
    public static void main(String[] args) {
        IntStream stream = new Random().ints(1, 7).limit(2);
        System.out.println(stream.max().getAsInt());
    }
}
```

Above code compiles and executes successfully and generates random integers.

Which of the following is not the possible output of above code?

A - 6

B - 4

C - 5

D - 7

Java Stream API - 55

Given code of Test.java file:

```
import java.time.LocalDate;
import java.util.Optional;
import java.util.stream.Stream;

public class Test {
    public static void main(String[] args) {
        Stream<LocalDate> stream = Stream.of(LocalDate.of(2018, 1,
         1), LocalDate.of(2018, 1, 1));
        Optional<LocalDate> optional = stream.distinct().findAny();

        System.out.println(optional.isPresent() + " : " +
         optional.get());
    }
}
```

What will be the result of compiling and executing Test class?

A - false : 2018-1-1

B - true : 2018-01-01

C - false : 2018-01-01

D - true : 2018-1-1

Java Stream API - 56

Given code of Test.java file:

```
import java.util.stream.Stream;

public class Test {
    public static void main(String[] args) {
        Stream<Double> stream = Stream.of(9.8, 2.3, -3.0);
        System.out.println(stream.min());
    }
}
```

What will be the result of compiling and executing Test class?

A - -3.0

B - Compilation error

C - Runtime Exception

D - 2.3

Java Stream API - 57

Given code of Test.java file:

```
import java.util.Arrays;

public class Test {
    public static void main(String[] args) {
        String [] names = {"Peter", "bonita", "John"};
        Arrays.stream(names).sorted((s1, s2) ->
         s1.compareToIgnoreCase(s2))
                .forEach(System.out::println);
    }
}
```

What will be the result of compiling and executing Test class?

A -

```
John
Peter
bonita
```

B -

```
bonita
John
Peter
```

C -

```
John
bonita
Peter
```

D -

```
Peter
bonita
John
```

Java Stream API - 58

Given code of Test.java file:

```
import java.util.stream.Stream;

class Employee {
    private String name;
    private double salary;

    public Employee(String name, double salary) {
        this.name = name;
        this.salary = salary;
    }

    public String getName() {
        return name;
    }

    public double getSalary() {
        return salary;
    }

    public String toString() {
        return "{" + name + ", " + salary + "}";
    }

    public static int salaryCompare(double d1, double d2) {
        return new Double(d2).compareTo(d1);
    }
}

public class Test {
    public static void main(String[] args) {
        Stream<Employee> employees = Stream.of(new Employee("Jack",
         10000),
                new Employee("Lucy", 12000), new Employee("Tom",
         7000));

        highestSalary(employees);
    }

    private static void highestSalary(Stream<Employee> emp) {
        System.out.println(emp.map(e ->
         e.getSalary()).max(Employee::salaryCompare));
    }
}
```

What will be the result of compiling and executing Test class?

A - Optional[10000.0]

B - Optional.empty

C - Optional[12000.0]

D - Optional[7000.0]

Java Stream API - 59

Given code of Test.java file:

```
import java.util.Arrays;
import java.util.stream.Stream;

public class Test {
    public static void main(String[] args) {
        String [] arr1 = {"Virat", "Rohit", "Shikhar", "Dhoni"};
        String [] arr2 = {"Bumrah", "Pandya", "Sami"};
        String [] arr3 = {};

        Stream<String[]> stream = Stream.of(arr1, arr2, arr3);
        stream.flatMap(s -> Arrays.stream(s)).sorted().forEach(s ->
         System.out.print(s + " "));
    }
}
```

What will be the result of compiling and executing Test class?

A - Bumrah Dhoni Pandya Rohit Sami Shikhar Virat

B - Virat Rohit Shikhar Dhoni Bumrah Pandya Sami

C - Virat Rohit Shikhar Dhoni Bumrah Pandya Sami null

D - Bumrah Dhoni Pandya Rohit Sami Shikhar Virat null

E - null Bumrah Dhoni Pandya Rohit Sami Shikhar Virat

Java Stream API - 60

Given:

```
import java.util.Arrays;
import java.util.List;
import java.util.function.UnaryOperator;

public class Test {
    public static void main(String[] args) {
        /* INSERT */
        List<Character> vowels = Arrays.asList('A', 'E', 'I', 'O',
         'U');
        vowels.stream().map(x ->
         operator.apply(x)).forEach(System.out::print); //Line n1
    }
}
```

Line n1 is causing compilation error as variable ‘operator’ is not found. Which of the following two options can replace /* INSERT */ such that output is: BFJPV?

A - UnaryOperator operator = c -> (char) c (c.charValue() + 1);

B - UnaryOperator operator = c -> c + 1;

C - UnaryOperator operator = c -> c + 1;

D - UnaryOperator operator = c -> c.charValue() + 1;

E - Function<Character, Character> operator = x -> (char)(x + 1);

F - Function<Character, Integer> operator = x -> x + 1;

Java Stream API - 61

Given code of Test.java file:

```
import java.util.stream.Stream;

public class Test {
    public static void main(String[] args) {
        Stream<Double> stream = Stream.generate(() -> new
         Double("1.0"));
        System.out.println(stream.sorted().findFirst());
    }
}
```

What will be the result of compiling and executing Test class?

A - Compilation error

B - Nothing is printed and program runs infinitely

C - Optional[1.0] is printed and program terminates successfully

D - Optional[1.0] is printed and program runs infinitely

Java Stream API - 62

Given code of Test.java file:

```
import java.util.stream.LongStream;

public class Test {
    public static void main(String[] args) {
        LongStream stream = LongStream.empty();
        System.out.println(stream.average());
    }
}
```

What will be the result of compiling and executing Test class?

A - Runtime exception

B - null

C - OptionalDouble.empty

D - 0.0

Java Stream API - 63

Given code of Test.java file:

```
import java.util.OptionalLong;

public class Test {
    public static void main(String[] args) {
        OptionalLong optional = OptionalLong.empty();
        System.out.println(optional.isPresent() + " : " +
         optional.getAsLong());
    }
}
```

What will be the result of compiling and executing Test class?

A - true : 0

B - true : null

C - Runtime Exception

D - false : 0

E - false : null

Java Stream API - 64

Given code of Test.java file:

```
import java.util.Optional;

public class Test {
    public static void main(String[] args) {
        Optional<Integer> optional = Optional.of(null);
        System.out.println(optional);
    }
}
```

What will be the result of compiling and executing Test class?

A - Optional[0]

B - NullPointerException is thrown at runtime

C - Optional.empty

D - Optional[null]

Java Stream API - 65

Given code of Test.java file:

```
import java.util.stream.Stream;

public class Test {
    public static void main(String[] args) {
        Stream.of(true, false, true).map(b -> b.toString()
                .toUpperCase()).peek(System.out::println);
    }
}
```

What will be the result of compiling and executing Test class?

A - Program executes successfully but nothing is printed on to the console

B -

```
true
false
true
```

C - Compilation error

D -

```
TRUE
FALSE
TRUE
```

Java Stream API - 66

Given code of Test.java file:

```
import java.util.Arrays;
import java.util.stream.Stream;

public class Test {
    public static void main(String[] args) {
        Stream<Double> stream = Arrays.asList(1.8, 2.2,
        3.5).stream();
        System.out.println(stream.reduce((d1, d2) -> d1 + d2));
        //Line 9
    }
}
```

What will be the result of compiling and executing Test class?

A - Line 9 causes Compilation error

B - Optional[7.5]

C - 7.5

Java Stream API - 67

Given code of Test.java file:

```
import java.util.Optional;
import java.util.stream.Stream;

public class Test {
    public static void main(String[] args) {
        Stream<String> stream = Stream.of("a", "as", "an", "and");
        Optional<String> first = stream.findFirst();
        if(first.ifPresent()) {
            System.out.println(first.get());
        }
    }
}
```

What will be the result of compiling and executing Test class?

A - Any element from the stream is printed

B - Compilation error

C - a

D - Runtime Exception