Pregunta 1

What will the following code print?

```
List<Integer> str = Arrays.asList(1,2, 3, 4 );

str.stream().filter(x->{

   System.out.print(x+" ");

   return x>2;

});
```

A. 1 2 3 4

B. 1 2 3 4 4

C. 4

D. It will not print anything

Pregunta 2

Given the following code:

```
import java.util.Arrays;

import java.util.List;

public class TestClass {

    public static void main(String[] args) {

        List<Integer> al = Arrays.asList(100, 200, 230, 291, 43);

        System.out.println( *INSERT CODE HERE* );

    }

}
```

Which of the following options will correctly print the number of elements that are less than 200?

A. al.asStream().reduce((i)->i<200).count();
B. al.stream().map((i)->i<200, i).count();
C. al.stream().filter((i)->i<200).list().count();
[✓ D. al.stream().filter((i)->i<200).count();]
E. al.asStream().filter((i)->i<200).count();

Pregunta 3

What will be the result of compilation and execution of the following code ?

```
DoubleStream is = DoubleStream.of(0, 2, 4); //1

double sum = is.filter( i->i%2 != 0 ).sum(); //2

System.out.println(sum); //3
```

A. It will print 0.0
B. It will print 6.0
C. It will print OptionalDouble[0.0] if line at //2 is replaced with OptionalDouble x = is.sum();
D. It will not compile.
E. It will throw an exception at run time.

Pregunta 4

What will the following code print when compiled and run?

```
List<String> values = Arrays.asList("Java EE", "C#", "Python");

boolean flag = values.stream().allMatch(str->{

    System.out.println("Testing: "+str);

    return str.equals("Java");

    });

System.out.println(flag);
```

A.
```
Testing: Java EE
false
```
B.
```
 Testing: Java EE
Testing: C#
Testing: Python
false
```
C.
```
Testing: Java EE
true
```
D. It will not compile because lambda expression is built incorrectly.

Pregunta 5

Given:

String sentence1 = "Carpe diem. Seize the day, boys. Make your lives extraordinary.";

String sentence2 = "Frankly, my dear, I don't give a damn!";

String sentence3 = "Do I look like I give a damn?";

List<String> sentences = Arrays.asList(sentence1, sentence2, sentence3);

Which of the following options will create a stream containing all the words in the three sentences without repetition?

A. Stream<String> strm = sentences.stream()

   .flatMap(str->Stream.of(str.split("[,.!?\r\n]")))

   .filter(s->s.length()>0)

   .distinct();

B. Stream<String> strm = sentences.stream()

   .map(str->Stream.of(str.split("[,.!?\r\n]")))

   .filter(s->s.length()>0)

   .distinct();

C. Stream<String> strm = sentences.stream()

   .forEach(str->Stream.of(str.split("[,.!?\r\n]")))

   .filter(s->s.length()>0)

   .distinct();

D. Stream<String> strm = sentences.stream()

   .flatMap(str-> str.split("[,.!?\r\n]"))

   .filter(s->s.length()>0)

   .merge();

E. Stream<String> strm = sentences.stream()

   .forEach(str->Stream.of(str.split("[,.!?\r\n]")))

   .filter(s->s.length()>0)

   .merge();