

Time left 0:07:35

**Question 1**

Not yet answered

Marked out of 1.00

Which snippet produces

```
Map<String, Integer> freq = {apple=2, banana=1}
```

- ☐ a. 

```
List<String> fruits = List.of("apple", "banana", "apple");  
fruits.stream()  
.collect(Collectors.groupingBy(f -> f, Collectors.counting()));
```
- ☐ b. 

```
List<String> fruits = List.of("apple", "banana", "apple");  
fruits.stream()  
.collect(Collectors.toMap(f->f, f->1, Integer::sum));
```
- ☐ c. 

```
List<String> fruits = List.of("apple", "banana", "apple");  
fruits.stream()  
.collect(Collectors.toMap(f->f, f->1));
```
- ☒ d. 

```
List<String> fruits = List.of("apple", "banana", "apple");  
fruits.stream()  
.collect(Collectors.groupingBy(f -> f, Collectors.summingInt(f -> 1)));
```

[Clear my choice](#)**Question 2**

Not yet answered

Marked out of 1.00

Predict the output

```
List<String> items = List.of("a", "b", "c");
```

```
String result = items.stream()
```

```
.reduce("", (a,b) -> a + b);
```

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

- ☒ a. abc
- ☐ b. cba
- ☐ c. [a, b, c]
- ☐ d. Compilation error

[Clear my choice](#)

**Question 3**

Not yet answered

Marked out of 1.00

Predict the output

```
List<Integer> nums = List.of(1,2,3,4);
```

```
int sum = nums.stream()
```

```
    .mapToInt(i -> i * 2)
```

```
    .sum();
```

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

- ☐ a. 10
- ☒ b. 20
- ☐ c. 16
- ☐ d. 8

[Clear my choice](#)**Question 4**

Not yet answered

Marked out of 1.00

Which snippet maps a list of lists to a flat list

```
[1,2,3,4]
```

```
from `List<List<Integer>> ll = List.of(List.of(1,2),List.of(3,4));`?
```

- ☐ a. 

```
ll.stream()  
    .flatMap(Collection::stream)  
    .collect(Collectors.toList());
```
- ☐ b. 

```
ll.stream()  
    .map(Collection::stream)  
    .flatMap(s->s)  
    .toList();
```
- ☒ c. Both A and B
- ☐ d. Neither

[Clear my choice](#)

**Question 5**

Not yet answered

Marked out of 1.00

Which snippet prints

Found: 42

only if any element equals 42?

- ☐ a. `Stream.of(10,42,30)`  
`.filter(i->i==42)`  
`.forEach(i->System.out.println("Found: "+i));`
- ☐ b. `List.of(10,42,30).stream()`  
`.anyMatch(i->i==42)`  
`.ifPresent(i->System.out.println("Found: "+i));`
- ☒ c. `Stream.of(10,42,30)`  
`.filter(i->i==42)`  
`.findFirst()`  
`.ifPresent(i->System.out.println("Found: "+i));`
- ☐ d. `Stream.of(10,42,30)`  
`.findAny(i->i==42)`  
`.ifPresent(i->System.out.println("Found: "+i));`

[Clear my choice](#)**Question 6**

Not yet answered

Marked out of 1.00

Predict the output

```
List<String> names = List.of("Anna","Bob","Alice");
```

```
long cnt = names.stream()
```

```
.filter(s -> s.startsWith("A"))
```

```
.count();
```

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

- ☐ a. 1
- ☒ b. 2
- ☐ c. 3
- ☐ d. 0

[Clear my choice](#)

**Question 7**

Not yet answered

Marked out of 1.00

Which snippet produces the product of all elements

24

```
from `List<Integer> nums = List.of(1,2,3,4);`?
```

- ☐ a. `nums.stream().reduce(1, (a,b)->a*b);`
- ☐ b. `nums.stream().reduce((a,b)->a*b).get();`
- ☒ c. Both A and B
- ☐ d. Neither

[Clear my choice](#)**Question 8**

Not yet answered

Marked out of 1.00

Predict the output

```
List<Integer> data = List.of(5,3,8,3,9,5);
```

```
data.stream()
```

```
.distinct()
```

```
.sorted()
```

```
.forEach(n -> System.out.print(n + " "));
```

- ☐ a. 5 3 8 9
- ☒ b. 3 5 8 9
- ☐ c. 5 3 8 3 9
- ☐ d. 3 8 5 9

[Clear my choice](#)

**Question 9**

Not yet answered

Marked out of 1.00

Which snippet produces a sorted, distinct list of even squares

[4,16,36]

from `List<Integer> nums = List.of(1,2,3,4,5,6);`?

- ☐ a. `nums.stream()  
 .map(i->i*i)  
 .filter(i->i%2==0)  
 .sorted()  
 .distinct()  
 .collect(Collectors.toList());`
- ☐ b. `nums.stream()  
 .filter(i->i%2==0)  
 .map(i->i*i)  
 .distinct()  
 .sorted()  
 .toList();`
- ☒ c. Both A and B
- ☐ d. Neither

[Clear my choice](#)**Question 10**

Not yet answered

Marked out of 1.00

Predict the output

```
Optional<String> first = Stream.<String>empty()  
    .findFirst();
```

```
System.out.println(first.isPresent());
```

- ☐ a. true
- ☒ b. false
- ☐ c. null
- ☐ d. NoSuchElementException

[Clear my choice](#)

**Question 11**

Not yet answered

Marked out of 1.00

Which snippet prints only the first two distinct names

Alice

Bob

from `List<String> names = List.of("Bob","Alice","Bob","Charlie");`?

- ☒ a. `names.stream()  
 .distinct()  
 .limit(2)  
 .forEach(System.out::println);`
- ☐ b. `names.stream()  
 .limit(2)  
 .distinct()  
 .forEach(System.out::println);`
- ☐ c. Both A and B
- ☐ d. Neither

[Clear my choice](#)**Question 12**

Not yet answered

Marked out of 1.00

Predict the output

```
List<String> vals = List.of("one","two","three");
```

```
String joined = vals.stream()
```

```
    .collect(Collectors.joining("|"));
```

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

- ☐ a. one,two,three
- ☒ b. one|two|three
- ☐ c. [one|two|three]
- ☐ d. one two three

[Clear my choice](#)

**Question 13**

Not yet answered

Marked out of 1.00

Which snippet prints

0 2 4 6 8

- ☐ a. `IntStream.rangeClosed(0,8)`  
`.filter(i->i%2==0)`  
`.forEach(System.out::print);`
- ☒ b. `IntStream.range(0,5)`  
`.map(i->i*2)`  
`.forEach(i->System.out.print(i+ " "));`
- ☐ c. `IntStream.of(0,2,4,6,8)`  
`.forEach(System.out::print);`
- ☐ d. `IntStream.range(0,8)`  
`.map(i->i*2)`  
`.forEach(i->System.out.print(i+ " "));`

[Clear my choice](#)**Question 14**

Not yet answered

Marked out of 1.00

Which snippet creates a comma-separated string

a,b,c

from `List<String> lst = List.of("a","b","c");`?

- ☐ a. `String s = String.join(",", lst);`
- ☐ b. `String s = lst.stream().collect(Collectors.joining(","));`
- ☐ c. `String s = lst.stream().reduce((x,y)-> x+","+y).get();`
- ☒ d. All of these

[Clear my choice](#)

**Question 15**

Not yet answered

Marked out of 1.00

Predict the output

```
IntStream.range(1,5)
    .skip(2)
    .forEach(System.out::print);
```

- ☐ a. 1234
- ☐ b. 12
- ☒ c. 345
- ☐ d. 34

[Clear my choice](#)**Question 16**

Not yet answered

Marked out of 1.00

Predict the output

```
List<Integer> list = List.of(1,2,3,4,5);
long even = list.parallelStream()
    .filter(i -> i % 2 == 0)
    .count();
System.out.println(even);
```

- ☐ a. 3
- ☒ b. 2
- ☐ c. 0
- ☐ d. Depends on thread scheduling

[Clear my choice](#)



**Question 17**

Not yet answered

Marked out of 1.00

Which snippet finds the maximum value 9  
from `List<Integer> nums = List.of(3,9,1,4);`?

- ☐ a. `nums.stream().max(Integer::compare).get();`
- ☐ b. `nums.stream().reduce(Integer::max).get();`
- ☒ c. Both A and B
- ☐ d. Neither

[Clear my choice](#)**Question 18**

Not yet answered

Marked out of 1.00

Predict the output

```
Stream.of("x","y","z")  
    .flatMap(s -> Stream.of(s.toUpperCase(), s.toLowerCase()))  
    .forEach(System.out::print);
```

- ☐ a. xyzXYZ
- ☐ b. XYZxyz
- ☒ c. XxYyZz
- ☐ d. xXyYzZ

[Clear my choice](#)

**Question 19**

Not yet answered

Marked out of 1.00

Predict the output

```
List<String> l = List.of(" a ", " b", "c ");  
l.stream()  
.map(String::trim)  
.forEach(s -> System.out.print "[" + s + " ]");
```

- ☐ a. [a][b][c]
- ☐ b. [ a ][ b][c ]
- ☐ c. [a, b, c]
- ☒ d. [ a ][ b ][ c ]

[Clear my choice](#)**Question 20**

Not yet answered

Marked out of 1.00

Which snippet yields

[A, B, C]

from `List<String> list = List.of("a","b","c");`?`

- ☐ a. `list.stream()  
.map(String::toUpperCase)  
.collect(Collectors.toList());`
- ☐ b. `list.stream()  
.collect(Collectors.toList())  
.stream()  
.map(String::toUpperCase)  
.toList();`
- ☐ c. `list.parallelStream()  
.map(String::toUpperCase)  
.toList();`
- ☒ d. All of these

[Clear my choice](#)