Lambda Built-in Functional Interfaces - 0

Given:

```
import java.util.function.Supplier;

class Document {
    void printAuthor() {
        System.out.println("Document-Author");
    }
}

class RFP extends Document {
    @Override
    void printAuthor() {
        System.out.println("RFP-Author");
    }
}

public class Test {
    public static void main(String[] args) {
        check(Document::new);
        check(RFP::new);
    }

    private static void check(_____________ supplier) {
        supplier.get().printAuthor();
    }
}
```

Given options to fill the blanks:

```
Supplier<Document>
Supplier<? extends Document>
Supplier<? super Document>
Supplier<RFP>
Supplier<? extends RFP>
Supplier<? super RFP>
Supplier
```

How many of the above options can fill the blank space, such that output is:

```
Document-Author
RFP-Author
```

A - Only two options

B - Only three options

C - Only one option

D - More than three options

Lambda Built-in Functional Interfaces - 1

Given code of Test.java file:

```
import java.util.function.Consumer;

public class Test {
    public static void main(String[] args) {
        Consumer<Integer> consumer = System.out::print;
        Integer i = 5;
        consumer.andThen(consumer).accept(i++); //Line 7
    }
}
```

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

A - 66

B - 55

C - 56

D - Compilation error

Lambda Built-in Functional Interfaces - 2

Given code of Test.java file:

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

interface StringConsumer extends Consumer<String> {
    @Override
    public default void accept(String s) {
        System.out.println(s.toUpperCase());
    }
}

public class Test {
    public static void main(String[] args) {
        StringConsumer consumer = s ->
         System.out.println(s.toLowerCase());
        List<String> list = Arrays.asList("Dr", "Mr", "Miss",
        "Mrs");
        list.forEach(consumer);
    }
}
```

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

A -

```
DR
MR
MISS
MRS
```

B -

```
dr
mr
miss
mrs
```

C - Runtime exception

D - Compilation error

Lambda Built-in Functional Interfaces - 3

Given code of Test.java file:

```
import java.util.NavigableMap;
import java.util.TreeMap;
import java.util.function.BiConsumer;

public class Test {
    public static void main(String[] args) {
        NavigableMap<Integer, String> map = new TreeMap<>();
        BiConsumer<Integer, String> consumer = map::putIfAbsent;
        consumer.accept(1, null);
        consumer.accept(2, "two");
        consumer.accept(1, "ONE");
        consumer.accept(2, "TWO");

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

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

A - {1=null, 2=two}

B - {1=null, 2=two}

C - {1=ONE, 2=TWO}

D - {1=null, 2=TWO}

E - {1=ONE, 2=two}

Lambda Built-in Functional Interfaces - 4

Given code of Test.java file:

```
import java.util.function.BiPredicate;

public class Test {
    public static void main(String[] args) {
        String [] arr = {"A", "ab", "bab", "Aa", "bb", "baba",
         "aba", "Abab"};
        BiPredicate<String, String> predicate = String::startsWith;

        for(String str : arr) {
            if(predicate.negate().test(str, "A"))
                System.out.println(str);
        }
    }
}
```

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

A -

```
A
Aa
Abab
```

B -

```
ab
bab
bb
baba
aba
```

C -

```
bab
bb
baba
```

D -

```
ab
aba
```

Lambda Built-in Functional Interfaces - 5

Given code of Test.java file:

```
import java.util.function.BiPredicate;

public class Test {
    public static void main(String[] args) {
        BiPredicate<String, String> predicate =
         String::equalsIgnoreCase;
        System.out.println(predicate.test("JaVa", "Java"));
    }
}
```

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

A - true

B - false

C - Runtime error

D - Compilation error

Lambda Built-in Functional Interfaces - 6

Given code of Test.java file:

```
import java.util.function.ToIntFunction;

public class Test {
    public static void main(String[] args) {
        String text = "Aa aA aB Ba aC Ca";
        ToIntFunction<String> func = text::indexOf;
        System.out.println(func.applyAsInt("a"));
    }
}
```

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

A - Compilation error

B - 0

C - 1

D - -1

Lambda Built-in Functional Interfaces - 7

Given code of Test.java file:

```
import java.util.LinkedHashMap;
import java.util.Map;
import java.util.function.BiConsumer;
import java.util.function.BiFunction;

public class Test {
    public static void main(String[] args) {
        Map<Integer, Integer> map = new LinkedHashMap<>();
        map.put(1, 10);
        map.put(2, 20);
        BiConsumer<Integer, Integer> consumer = (k, v) -> {
            System.out.println(k + ":" + v);
        };

        BiFunction<Integer, Integer, Integer> function = (k, v) -> {
            System.out.println(k + ":" + v);
            return null;
        };
        //Line n1
    }
}
```

Which of the following options will replace //Line n1 such that below output is printed to the console?

```
1:10
2:20
```

A - map.forEach(consumer);

B - map.forEach(function);

C - map.forEachOrdered(function);

D - map.forEachOrdered(consumer);

Lambda Built-in Functional Interfaces - 8

Given code of Test.java file:

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

interface StringConsumer extends Consumer<String> {
    @Override
    public default void accept(String s) {
        System.out.println(s.toUpperCase());
    }
}

public class Test {
    public static void main(String[] args) {
        StringConsumer consumer = new StringConsumer() {
            @Override
            public void accept(String s) {
                System.out.println(s.toLowerCase());
            }
        };
        List<String> list = Arrays.asList("Dr", "Mr", "Miss",
        "Mrs");
        list.forEach(consumer);
    }
}
```

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

A - Runtime exception

B - Compilation error

C -

```
DR
MR
MISS
MRS
```

D -

```
dr
mr
miss
mrs
```

Lambda Built-in Functional Interfaces - 9

Given code of Test.java file:

```
import java.util.function.BiFunction;

public class Test {
    public static void main(String[] args) {
        BiFunction<Integer, Integer, Character> compFunc = (i, j) -
        > i + j;
        System.out.println(compFunc.apply(0, 65));
    }
}
```

NOTE: ASCII value of A is 65.

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

A - A

B - 65

C - Compilation error

Lambda Built-in Functional Interfaces - 10

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> list = Arrays.asList("north", "east", "west",
         "south");
        list.replaceAll(s ->
         s.substring(0,1).toUpperCase().concat(s.substring(1)));

        System.out.println(list);
    }
}
```

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

A - [N, E, W, S]

B - [north, east, west, south]

C - [n, e, w, S]

D - [North, East, West, South]

E - [NORTH, EAST, WEST, SOUTH]

Lambda Built-in Functional Interfaces - 11

Given code of Test.java file:

```
import java.util.*;
import java.util.function.DoublePredicate;

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 void setSalary(double salary) {
        this.salary = salary;
    }

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

public class Test {
    public static void main(String[] args) {
        List<Employee> employees = Arrays.asList(new
         Employee("Jack", 8000),
                new Employee("Lucy", 12000));
        updateSalary(employees, d -> d < 10000);
        employees.forEach(System.out::println);
    }

    private static void updateSalary(List<Employee> list,
         DoublePredicate predicate) {
        for(Employee e : list) {
            if(predicate.negate().test(e.getSalary())) {
                e.setSalary(e.getSalary() + 2000);
            }
        }
    }
}
```

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

A -

```
{Jack, 8000.0}
{Lucy, 14000.0}
```

B -

```
{Jack, 10000.0}
{Lucy, 14000.0}
```

C -

```
{Jack, 10000.0}
{Lucy, 12000.0}
```

D -

```
{Jack, 8000.0}
{Lucy, 12000.0}
```

Lambda Built-in Functional Interfaces - 12

Given code of Test.java file:

```
import java.util.Date;
import java.util.function.*;

public class Test {
    public static void main(String[] args) {
        /*INSERT*/ obj = Date::new;//Constructor reference for Date()
         constructor
        Date date = obj.get(); //Creates an instance of Date class.
        System.out.println(date);
    }
}
```

Which of the following options can replace /*INSERT*/ such that on executing Test class, current date and time is displayed in the output?

A - Function

B - Function

C - Supplier

D - Function

E - Supplier

F - Supplier

Lambda Built-in Functional Interfaces - 13

Given code of Test.java file:

```
import java.util.function.BiFunction;

public class Test {
    public static void main(String[] args) {
        BiFunction<String, String, String> func = String::concat;
        System.out.println(func.apply("James", "Gosling"));
    }
}
```

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

A - Gosling

B - JamesGosling

C - Gosling James

D - James

E - GoslingJames

F - James Gosling

Lambda Built-in Functional Interfaces - 14

Consider below code:

```
import java.util.function.Function;

public class Test {
    public static void main(String[] args) {
        Function<Integer, Integer> f = x -> x + 10;
        Function<Integer, Integer> g = y -> y * y;

        Function<Integer, Integer> fog = g.compose(f); //Line 8
        System.out.println(fog.apply(10));
    }
}
```

On execution, Test class prints 400 on to the console. Which of the statements can replace Line 8 such that there is no change in the output?

A - Function fog = f.compose(g);

B - Function fog = f.andThen(g);

C - Function fog = g.andThen(f);

Lambda Built-in Functional Interfaces - 15

Given code of Test.java file:

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

public class Test {
    public static void main(String[] args) {
        List<Integer> list = Arrays.asList(10, 100, 1000);
        list.replaceAll(i -> -i++);

        System.out.println(list);
    }
}
```

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

A - Compilation error

B - [-10, -100, -1000]

C - [-11, -101, -1001]

D - [-9, -99, -999]

E - [10, 100, 1000]

Lambda Built-in Functional Interfaces - 16

Which of the following is the only Functional interface available for boolean primitive type?

A - BooleanPredicate

B - BooleanFunction

C - BooleanSupplier

D - BooleanConsumer

Lambda Built-in Functional Interfaces - 17

Given code of Test.java file:

```
import java.util.function.BiFunction;
import java.util.function.BiPredicate;

public class Test {
    public static void main(String[] args) {
        BiPredicate<String, String> predicate = String::contains;
        BiFunction<String, String, Boolean> func = (str1, str2) -> {
            return predicate.test(str1, str2) ? true : false;
        };

        System.out.println(func.apply("Tomato", "at"));
    }
}
```

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

A - Compilation error

B - null

C - false

D - true

Lambda Built-in Functional Interfaces - 18

Given code of Test.java file:

```
import java.util.function.Supplier;

public class Test {
    public static void main(String[] args) {
        Supplier<StringBuilder> supplier = () -> new
         StringBuilder(" olleH")
                .reverse().append("!dlroW").reverse();
        System.out.println(supplier.get());
    }
}
```

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

A - > olleHWorld!<

B - > olleH!dlrow<

C - >Hello World!<

D - >World!Hello <

E - >World! olleH<

Lambda Built-in Functional Interfaces - 19

Given code of Test.java file:

```
import java.util.function.Consumer;

class Counter {
    static int count = 1;
}
public class Test {
    public static void main(String[] args) {
        Consumer<Integer> add = i -> Counter.count += i;
        Consumer<Integer> print = System.out::println;
        add.andThen(print).accept(10); //Line 10
    }
}
```

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

A - 1

B - Compilation error

C - 11

D - 10

Lambda Built-in Functional Interfaces - 20

Given code of Test.java file:

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

public class Test {
    public static void main(String[] args) {
        List<Integer> list = Arrays.asList (2, 3, 4);
        UnaryOperator<Long> operator = s -> s*s*s;
        list.replaceAll(operator);
        list.forEach(System.out::println);
    }
}
```

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

A -

```
8
27
64
```

B -

```
2
3
4
```

C - Compilation error

D - Runtime exception

Lambda Built-in Functional Interfaces - 21

Given code of Test.java file:

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

public class Test {
    public static void main(String[] args) {
        LongStream.rangeClosed(51,75).filter(l -> l % 5 == 0)
                .forEach(l -> System.out.print(l + " "));
    }
}
```

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

A - 55 60 65 70 75

B - 75 70 65 60 55

C - 55 60 65 70

D - 75 70 65 60

Lambda Built-in Functional Interfaces - 22

You have to create below functional interface:

```
interface Generator<T, U> {
    U generate(T t);
}
```

Which of the following built-in interface you can use instead of above interface?

A - Supplier

B - Consumer

C - Function

D - Predicate

Lambda Built-in Functional Interfaces - 23

Built-in functional interfaces are part of which java package?

A - java.util.function

B - java.util

C - java.lang

D - java.function

Lambda Built-in Functional Interfaces - 24

Given code of Test.java file:

```
import java.util.function.BiFunction;
import java.util.function.BiPredicate;

public class Test {
    public static void main(String[] args) {
        BiFunction<String, String, String> func = (str1, str2) -> {
            return (str1 + str2);
        };

        BiPredicate<String, String> predicate = (str1, str2) -> {
            return func.apply(str1, str2).length() > 10;
        };

        String [] arr = {"vention", "historic", "sident",
         "sentation", "vious"};

        for(String str : arr) {
            if(predicate.test("pre", str)) {
                System.out.println(func.apply("pre", str));
            }
        }
    }
}
```

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

A -

```
prevention
prehistoric
president
presentation
```

B - Program terminates successfully without printing anything on to the console

C -

```
prevention
prehistoric
president
presentation
previous
```

D -

```
prevention
```

`prehistoric`

E - presentation

Lambda Built-in Functional Interfaces - 25

Given code of Test.java file:

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

public class Test {
    public static void main(String[] args) {
        String [] arr = {"A", "ab", "bab", "Aa", "bb", "baba",
         "aba", "Abab"};

        Predicate<String> p1 = s -> s.startsWith("A");
        Predicate<String> p2 = s -> s.startsWith("a");
        Predicate<String> p3 = s -> s.length() >= 3;

        processStringArray(arr, p1.or(p2).and(p3));
    }

    private static void processStringArray(String [] arr,
                                           Predicate<String>
         predicate) {
        for(String str : arr) {
            if(predicate.test(str)) {
                System.out.println(str);
            }
        }
    }
}
```

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

A -

```
aba
Abab
```

B -

```
bab
baba
aba
Abab
```

C -

```
A
ab
Aa
aba
Abab
```

D - Abab

Lambda Built-in Functional Interfaces - 26

Given code of Test.java file:

```
import java.util.function.Consumer;

public class Test {

    public static void main(String[] args) {
        Consumer<String> c1 = str -> {
            System.out.println(new StringBuilder(str).reverse()
                                        .toString().substring(2));
        };
        c1.accept("!yppahnu");
    }
}
```

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

A - happy!

B - unhapp

C - lyppah

D - ppahnu

Lambda Built-in Functional Interfaces - 27

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

```
import java.util.function.Function;

public class Test {
    public static void main(String[] args) {
        Function<char [], String> obj = String::new; //Line 5
        String s = obj.apply(new char[] {'j', 'a', 'v', 'a'}); //Line
         6
        System.out.println(s);
    }
}
```

A - Compilation error at Line 6

B - java

C - Compilation error at Line 5

D - Exception is thrown at runtime

Lambda Built-in Functional Interfaces - 28

Given code of Test.java file:

```
import java.util.function.Function;

public class Test {
    public static void main(String[] args) {
        Function<String, Integer> f1 = Integer::new;
        Function<String, String> f2 = s -> new
         StringBuilder(s).reverse().toString();
        System.out.println(f1.compose(f2).apply("12345"));
    }
}
```

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

A - 54321

B - NumberFormatException is thrown at runtime

C - 12345

D - Compilation error

Lambda Built-in Functional Interfaces - 29

Given code of Test.java file:

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

public class Test {
    public static void main(String[] args) {
        String [] arr = {"*", "**", "***", "****", "*****",
        "******"};
        Predicate<String> pr1 = s -> s.length() > 3;
        print(arr, pr1.negate());
    }

    private static void print(String [] arr, Predicate<String>
        predicate) {
        for(String str : arr) {
            if(predicate.test(str)) {
                System.out.println(str);
            }
        }
    }
}
```

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

A -

```
*
**
```

B -

```
****
*****
******
```

C -

```
*
**
***
****
*****
******
```

D -

```
*
**
***
```

Lambda Built-in Functional Interfaces - 30

Given code of Test.java file:

```
import java.util.function.BiFunction;

public class Test {
    public static void main(String[] args) {
        BiFunction<String, String, String> func = (s1, s2) ->
         s2.concat(s1).trim();
        System.out.println(func.apply(" CD", " AB"));
    }
}
```

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

A - CDAB

B - ABCD

C - CD AB

D - AB CD

Lambda Built-in Functional Interfaces - 31

Given code of Test.java file:

```
import java.util.function.DoubleFunction;
import java.util.function.DoubleUnaryOperator;

public class Test {
    public static void main(String[] args) {
        DoubleFunction<DoubleUnaryOperator> func = m -> n -> m + n;
        //Line n1
        System.out.println(func.apply(11).applyAsDouble(24)); //Line
        n2
    }
}
```

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

A - 22.0

B - Line n2 causes compilation error

C - 48.0

D - Line n1 causes compilation error

E - 35.0

Lambda Built-in Functional Interfaces - 32

Which of the following pairs correctly represent the Functional interface and its single abstract method?

A -

```
Consumer : apply
Function : accept
Supplier : test
Predicate : get
```

B -

```
Consumer : accept
Function : apply
Supplier : get
Predicate : test
```

C -

```
Consumer : accept
Function : apply
Supplier : test
Predicate : get
```

D -

```
Consumer : apply
Function : accept
Supplier : get
Predicate : test
```

Lambda Built-in Functional Interfaces - 33

Given code of Test.java file:

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

public class Test {
    public static void main(String[] args) {
        UnaryOperator<String> opr = s -> s.toString().toUpperCase();
         //Line n1
        System.out.println(opr.apply(new StringBuilder("Hello")));
         //Line n2
    }
}
```

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

A - Hello

B - Compilation error at Line n2

C - HELLO

D - Compilation error at Line n1

Lambda Built-in Functional Interfaces - 34

Given code of Test.java file:

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

public class Test {
    public static void main(String[] args) {
        String [] arr = {"A", "ab", "bab", "Aa", "bb", "baba",
         "aba", "Abab"};

        processStringArray(arr, /*INSERT*/);
    }

    private static void processStringArray(String [] arr,
         Predicate<String> predicate) {
        for(String str : arr) {
            if(predicate.test(str)) {
                System.out.println(str);
            }
        }
    }
}
```

Which of the following options can replace /*INSERT*/ such that on executing Test class all the array elements are displayed in the output? Select ALL that apply.

A - p -> true

B - p -> p.length() < 10

C - p -> !false

D - p -> p.length() >= 1

Lambda Built-in Functional Interfaces - 35

Given code of Test.java file:

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

public class Test {
    public static void main(String[] args) {
        IntConsumer consumer = i -> i * i * i;
        int result = IntStream.range(1, 5).sum();
        System.out.println(result);
    }
}
```

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

A - Compilation error

B - 225

C - 100

D - Runtime Exception

Lambda Built-in Functional Interfaces - 36

Which of the import statements correctly imports the functional interface Comparator?

A - import java. function.Comparator;

B - import java.util.Comparator;

C - import java.util. function.Comparator;

D - import java. lang.Comparator;

Lambda Built-in Functional Interfaces - 37

Given code of Test.java file:

```
import java.util.function.BiFunction;

public class Test {
    public static void main(String[] args) {
        BiFunction<Double, Double, Integer> compFunc =
         Double::compareTo;
        System.out.println(compFunc.apply(10.01, 11.99));
    }
}
```

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

A - -2

B - 2

C - 1

D - 0

E - -1

F - Compilation error

Lambda Built-in Functional Interfaces - 38

Given code of Test.java file:

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

public class Test {
    public static void main(String[] args) {
        LongFunction<LongUnaryOperator> func = a -> b -> b - a;
        //Line n1
        System.out.println(calc(func.apply(100), 50)); //Line n2
    }

    private static long calc(LongUnaryOperator op, long val) {
        return op.applyAsLong(val);
    }
}
```

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

A - 100

B - -100

C - Line n1 causes compilation error

D - -50

E - Line n2 causes compilation error

F - 50

Lambda Built-in Functional Interfaces - 39

Given code of Test.java file:

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

public class Test {
    public static void main(String[] args) {
        printNumbers(i -> i % 2 != 0);
    }

    private static void printNumbers(Predicate<Integer> predicate)
        {
        for(int i = 1; i <= 10; i++) {
            if(predicate.test(i)) {
                System.out.print(i);
            }
        }
    }
}
```

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

A - 246810

B - 1357911

C - 13579

D - 1234567891011

E - 12345678910

Lambda Built-in Functional Interfaces - 40

Given code of Test.java file:

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

public class Test {
    public static void main(String[] args) {
        final String password = "Oracle";
        UnaryOperator<String> opr1 = s -> s.replace('a', '@');
        //Line n1
        UnaryOperator<String> opr2 = s -> password.concat(s); //Line
        n2
        System.out.println("Password: " +
        opr1.apply(opr2.apply("!"))); //Line n3
    }
}
```

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

A - Compilation error at Line n1

B - Compilation error at Line n2

C - Password: Or@cle!

D - Password: Oracle!

E - Compilation error at Line n3