2. Given the definition of `Doc` class:

```
package p1;
public abstract sealed class Doc permits WordDoc {
    String docName;
    public abstract void printDoc();
    public String getName() { return docName;}
}
```

Which statement is true?

- ○ The `WordDoc` class should exist within `p1` of `module1`.
- ◉ The `WordDoc` class could exist within another package, `p2`, of `module1`.
- ○ The `Doc` class must declare non-`abstract`.
- ○ The `getName()` method should be `abstract`.

3. Given:

```
final class Folder {    // line n1
    // line n2
    public void open(){
        System.out.print("Open ");
    }
}

public class Test {
    public static void main(String[] args) throws Exception {
        try (Folder f = new Folder()) {
            f.open();
        }
    }
}
```

```
public class Test {
    public static void main(String[] args) throws Exception {
        try (Folder f = new Folder()) {
            f.open();
        }
    }
}
```

Which two modifications enable the code to print `Open Close`?

- [ ] Replace `line n1` with:
  `class Folder extends Closeable {`
- [x] At `line n2`, insert:
  ```
  public void close() throws IOException {
  System.out.print("Close ");
  }
  ```
- [ ] At `line n2`, insert:
  ```
  final void close() {
  System.out.print("Close ");
  }
  ```
- [x] Replace `line n1` with:
  `class Folder implements AutoCloseable {`
- [ ] Replace `line n1` with:
  `class Folder extends Exception {`

4. Given:

```
class StockException extends Exception {
    public StockException(String s) { super(s); }
}
class OutofStockException extends StockException {
    public OutofStockException(String s) { super(s); }
}
```

and the code fragment:

```
public class Test {
    public static void main(String[] args) throws OutofStockException {
        m();
    }
    public static void m() throws OutofStockException {
        try {
            throw new StockException("Raised.");
        } catch (Exception e) {
            throw new OutofStockException(e.getMessage());
        }
    }
}
```

Which statement is true?

```
class OutofStockException extends StockException {
    public OutofStockException(String s) { super(s); }
}
```

and the code fragment:

```
public class Test {
    public static void main(String[] args) throws OutofStockException {
        m();
    }
    public static void m() throws OutofStockException {
        try {
            throw new StockException("Raised.");
        } catch (Exception e) {
            throw new OutofStockException(e.getMessage());
        }
    }
}
```

Which statement is true?

- ○ The program throws `StockException`.
- ◉ The program throws `OutofStockException`.
- ○ The program fails to compile.
- ○ The program throws `ClassCastException`.

6. Given:

```
public class Weather {
      public enum Forecast {
            SUNNY, CLOUDY, RAINY;
            @Override
            public String toString() { return "SNOWY";}
      }

      public static void main(String[] args) {
         System.out.print(Forecast.SUNNY.ordinal() + " ");

System.out.print(Forecast.valueOf("cloudy".toUpperCase()));
      }
}
```

What is the result?

- 1 RAINY
- Compilation fails
- 0 CLOUDY
- 1 SNOWY
- (●) 0 SNOWY

7. Given:

```
public enum Design {
    CEO('A'), CMO('B'), CTO('C'), CFO('D');
    char c;
    private Design(char c) {
        this.c = c;
    }
}
```

and the code fragment:

```
Arrays.stream(Design.values()).dropWhile(s -> s.equals(Design.CMO));
switch (Design.valueOf("CMO")) {
    case CEO -> System.out.println("Executive");
    case CMO -> System.out.println("Marketing");
    case CFO -> System.out.println("Finance");
    case CTO -> System.out.println("Technical");
    default -> System.out.println("UnDefined");
}
```

What is the result?

```
    char c;
    private Design(char c) {
        this.c = c;
    }
}
```

and the code fragment:

```
Arrays.stream(Design.values()).dropWhile(s -> s.equals(Design.CMO));
switch (Design.valueOf("CMO")) {
    case CEO -> System.out.println("Executive");
    case CMO -> System.out.println("Marketing");
    case CFO -> System.out.println("Finance");
    case CTO -> System.out.println("Technical");
    default -> System.out.println("UnDefined");
}
```

What is the result?

- ○ Marketing
  UnDefined
- ○ UnDefined
- ○ Marketing
  Finance
  Technical
- ◉ Marketing

9. Which statement is true about migration?

- (●) Every module is moved to the module path in a top-down migration.
- ( ) Unnamed modules are automatic modules in a top-down migration.
- ( ) Every module is moved to the module path in a bottom-up migration.
- ( ) The required modules migrate before the modules that depend on them in a top-down migration.

12. Given the code fragment:

```
class Book {
     String author;
     String title;
     Book(String authorName, String title) {
          this.author = authorName;
          this.title = title;
     }
}

class SortBook {
     public static void main(String[] args) {
          List<Book> books = List.of(new Book("A1","T1"), new Book("A2", "T2"), new Book("A1","T2")); // Line n1
          books.sort((Book a, Book b) -> a.title.compareTo(b.title));         // Line n2
          System.out.println(books);
     }
}
```

Which action sorts the book list?

- ○ At `Line n1`, convert `books` type to mutable array type.
- ○ At `Line n2`, replace `compareTo()` with `compare()`.
- ◉ At `Line n1`, convert `books` type to mutable `ArrayList` type.
- ○ At `Line n2`, replace `books.sort()` with `books.stream().sort()`.

13. Given the code fragment:

```
List<String> lst = new ArrayList<String>();
lst.add("e1");
lst.add("e3");
lst.add("e2");

int x1 = Collections.binarySearch(lst, "e3");
System.out.println(x1);
Collections.sort(lst);
int x2 = Collections.binarySearch(lst, "e3");
System.out.println(x2);

Collections.reverse(lst);
int x3 = Collections.binarySearch(lst, "e3");
System.out.println(x3);
```

What is the result?

```
int x1 = Collections.binarySearch(lst, "e3");
System.out.println(x1);
Collections.sort(lst);
int x2 = Collections.binarySearch(lst, "e3");
System.out.println(x2);

Collections.reverse(lst);
int x3 = Collections.binarySearch(lst, "e3");
System.out.println(x3);
```

What is the result?

- ○
  ```
  0
  2
  -2
  ```
- ○
  ```
  2
  2
  0
  ```
- ○
  ```
  1
  1
  1
  ```
- ◉
  ```
  1
  2
  -4
  ```

14. Given:

```
package com.transport.vehicle.cars;

public interface Car {
     int getSpeed();
}
```

and

```
package com.transport.vehicle.cars.impl;

import com.transport.vehicle.cars.Car;

public class CarImpl implements Car {
   private int speed;

   public CarImpl() {
      this(10);
   }

   public CarImpl (int speed) {
      this.speed = speed;
   }

    @Override
    public int getSpeed() {
       return speed;
   }
```

Which two should the `module-info` file include for it to represent the service provider interface?

- [ ] `requires com.transport.vehicle.cars.Car;`
- [ ] `requires com.transport.vehicle.cars;`
- [ ] `provides com.transport.vehicle.cars.impl.CarImpl to com.transport.vehicle.cars.Car;`
- [x] `exports com.transport.vehicle.cars;`
- [ ] `exports com.transport.vehicle;`
- [x] `provides com.transport.vehicle.cars.Car with com.transport.vehicle.cars.impl.CarImpl;`

15. Which two statements are true about modules in the Java Platform Module System?

- [x] Packages within the module are hidden by default.
- [ ] The classes in a module are loaded using class-path.
- [ ] The `module-info.java` has to be located in the Java `bin` directory.
- [ ] They support versioning of Java modules.
- [x] They are a group of related Java packages and resources such as XML files, images, and properties file.

16. Given:

```
public class App {
    public static void main(String[] args) {
        Doc ts = new Doc("Sales");
        ts.printDoc();
        Doc ts1 = new Doc("Purchase");
        ts1.printDoc();
    }
}

class Doc {
    public static Integer dId = 100;
    String name;
    Doc(String n) {
        this.dId = ++dId;
        this.name = n;
    }
    public void printDoc() {
        System.out.println(name + " - " + dId + " is printed.");
    }
}
```

17. Which two record definitions are valid?

- [x]
```
record Item(int id, String name) {
    public Item {
        if (name.length() < 3) {
            throw new IllegalArgumentException();
        }
    }
    static String qualityCode;
}
```

- [x]
```
record Item(int id, String name) {
    static {
        if (name.length() < 3) {
            throw new IllegalArgumentException();
        }
        String qualityCode;
    }
}
```

- [ ]
```
record Item(int id, String name) {
    public Item {
        if (name.length() < 3) {
            throw new IllegalArgumentException();
        }
    }
    String qualityCode;
}
```

18. Given the code fragment:

```
int a = 2;
int b = ~a;
int c = a^b;
boolean d = a < b & a > c++;
System.out.println(d + " " + c);
boolean e = a > b && a > c++;
System.out.println(e + " " + c);
```

What is the result?

- ○ false 1
  true 2
- ○ false 1
  false 2
- ◉ false 0
  true 2
- ○ true 1
  false 2

19. Given the code fragment:

```
String s = "10_00";
Integer s2 = 10_00;
// Line n1
System.out.println(res);
```

Which two statements at `Line n1` independently enable you to print `1250`?

- [ ] `Integer res = 250 + Integer.parseInt(s);`
- [x] `Integer res = 250;`
  `res =+ s2;`
- [ ] `Integer res = 250 + Integer(s2);`
- [ ] `Integer res = 250 + Integer.valueOf(s);`
- [x] `Integer res = 250 + s2;`
- [ ] `Integer res = 250 + s;`

20. Which two interface definitions compile?

- [x]
```
interface IFace1 {
    public static final String sName = "House and Mouse.";
    private void readStory() {}
}
```

- [ ]
```
interface IFace3 {
    public default int getCounter() { return 10; }
    public final void displayTitle() { System.out.println("Displayed."); }
}
```

- [x]
```
public abstract sealed interface SealedInterface permits Story, Art {
    default String getTitle() { return "Book Title" ; }
}
class Story implements Book {}
class Art implements Book {}
```

- [ ]
```
interface IFace4 {
    public default int getCounter1() { return 10; }
    public void displayTitle1();
}
```

- [ ]
```
interface IFace2 {
    public default int getCounter() { return 10; }
    public static void displayTitle();
    public void shareFile();
}
```

21. Given:

```
interface IFace {
    public void m1();
    public default void m2() {
        System.out.println("m2");
    }
    public static void m3() {
        System.out.println("m3");
    }
    private void m4() {
        System.out.println("m4");
    }
}

class MyC implements IFace {
    public void m1() {
        System.out.println("Hello");
    }
}
```

Which two method invocations execute?

```
    public static void m3() {
        System.out.println("m3");
    }
    private void m4() {
        System.out.println("m4");
    }
}

class MyC implements IFace {
    public void m1() {
        System.out.println("Hello");
    }
}
```

Which two method invocations execute?

- [ ] `IFace myClassObj = new MyC();`
  `myClassObj.m3();`
- [ ] `IFace myClassObj = new MyC();`
  `myClassObj.m4();`
- [x] `new MyC().m2();`
- [ ] `IFace.m4();`
- [x] `IFace.m2();`
- [ ] `IFace.m3();`

22. Both the `dataSrc.txt` file and the `dataBkup.txt` file exist,

and the code fragment:

```
try ( FileInputStream fis = new FileInputStream("dataSrc.txt");
      FileOutputStream fos = new FileOutputStream("dataBkup.txt");) {
      // Line n1
} catch (IOException e) {
      System.out.println(e);
}
```

Which code fragment at `Line n1` copies the content from `dataSrc.txt` to `dataBkup.txt`?

- ○ `Files.copy(fis, fos, StandardCopyOption.REPLACE_EXISTING);`
- ○ `Files.copy(fis, fos);`
- ◉ `fis.transferTo(fos);`
- ○ `fos.transferFrom(fis);`

23. Given the content of the `in.txt` file:

```
0123456789
```

and the code fragment:

```
char[] buffer = new char[8];
int count = 0;
try(FileReader in = new FileReader("in.txt");
    FileWriter out = new FileWriter("out.txt")) {
        while((count = in.read(buffer)) != -1) {
            out.write(buffer);
        }
}
```

What is the content of the `out.txt` file?

- ○ 01234567
- ○ 0123456789
- ○ 012345678 1234
- ◉ 0123456789234567
- ○ 012345678901234
- ○ 01234567 8

25. Given the code fragment:

```
// Login time:2021-01-12T21:58:18.817Z
Instant loginTime = Instant.now();
Thread.sleep(1000);

// Logout time:2021-01-12T21:58:19.880Z
Instant logoutTime = Instant.now();

loginTime = loginTime.truncatedTo(ChronoUnit.MINUTES);    // line n1
logoutTime = logoutTime.truncatedTo(ChronoUnit.MINUTES);

if (logoutTime.isAfter(loginTime))
    System.out.println("Logged out at: " + logoutTime);
else
    System.out.println("Can't logout");
```

What is the result?

- (●) `Can't logout`
- ( ) A compilation error occurs at line n1.
- ( ) `Logged out at: 2021-01-12T21:58:19.880Z`
- ( ) `Logged out at: 2021-01-12T21:58:00Z`

26. Given the code fragment:

```
Duration duration = Duration.ofMillis(5000);
System.out.print(duration);
duration = Duration.ofSeconds(60);
System.out.print(duration);
Period period = Period.ofDays(6);
System.out.print(period);
```

What is the result?

- ○ PT5000SPT60MP6D
- ◉ PT5SPT1MP6D
- ○ 5S1M6D
- ○ 5000S60M6D

27. Given:

```
public class Test {
   public static void main(String[] args) {
        List<String> elements =
              Arrays.asList("car", "truck", "car",
                            "bicycle", "car", "truck", "motorcycle");
        Map<String, Long> outcome =

 elements.stream().collect(Collectors.groupingBy(Function.identity(), Collectors.counting() ) );
        System.out.println(outcome);
   }
}
```

What is the result?

- (●) {bicycle=1, car=3, motorcycle=1, truck=2}
- ( ) {3:bicycle, 0:car, 6:motorcycle, 5:truck}
- ( ) {bicycle, car, motorcycle, truck}
- ( ) Compilation fails.
- ( ) {bicycle=7, car=7, motorcycle=7, truck=7}

28. Given the code fragment:

```
Stream<String> s1 = Stream.of("A", "B", "C", "B");
Stream<String> s2 = Stream.of("A", "D", "E");
Stream.concat(s1, s2).parallel().distinct().forEach(element -> System.out.print(element));
```

What is the result?

- (●) ABCDE // the order of elements is unpredictable
- ( ) ABBCDE // the order of elements is unpredictable
- ( ) ABCDE
- ( ) ADEABCB // the order of elements is unpredictable