1. What is wrong with this code?

```
public void Foo ()  }
        try {
          lock() // lock some resource
          // open some resource
          // try to change some things
          // fool around a bit

        }
        catch (Exception e)   {
        }
        catch (ConcurrentModificationException e) {

        System.out.println(„ bad stuff going on today!“)
        }
        finally {
                return;
        }
}
```

The second catch-block will be never reached, because Exception is the parent class, so all subclasses that inheritance Exception will be never caught if wrote after Exception-Catch.

2. What will be the output of the program?

```
public class TestException
{
   public static void badCall()
   {
      System.out.print("throwing it ");
      throw new RuntimeException();
   }
   public static void main(String [] args)
   {
      try
      {
         System.out.print("hello ");
         badCall();
      }
      catch (Exception re )
      {
         System.out.print("caught ");
      }
      finally
      {
         System.out.print("finally ");
      }
      System.out.println("after ");
   }
```

Hello throwing it caught finally after

3. Output?

```
public class TestException1
{
        public static void main(String [] args)
        {
           try
           {
              badMethod();
              System.out.print("A");
           }
           catch (Exception ex)
           {
              System.out.print("B");
           }
           finally
           {
              System.out.print("C");
           }
           System.out.print("D");
        }
        public static void badMethod()
        {
           throw new Error();
        }
}
```

This won't compile

4. Make it compile!

```
public class TestException2
        {
           class TestException extends Exception {}  // inner class
                   public void runTest() throws TestException {}

                   public void test() throws TestException
                      {
                         runTest();
                      }
        }
```

5. When should you re-throw a caught exception?

If you don't want to have the responsibility to handle the Error in this Method and another class or Method is responsible to handle Errors. For example, an API is throwing Errors, which YOU must handle and not the API.

6. A banking software detects, that a certain customer ID is not in the database.
Is this a) a system exception, b) a custom exception, c) no exception.

a) Or b), because you handle it as IllegalArgumentException or create own Exception for example NoCustomerIdException. I would suggest the second version because you can handle it like you want. For example, don't handle it or just put out a message.

7. Was ist der Vorteil von Exceptions gegenüber dem Auswerten von Fehlerwerten im Return?

Clean Coding, and you can decide when you want to handle it. Also, maybe you aren't able to evaluate the Error so you have to rethrow it to someone who is able to or is responsible for it. Like that you are able to work on and you don't get interrupted in your work.