1. (2 pts) Error, RuntimeException, and their subclasses are called **unchecked** exceptions, and Throwable and all other subclasses of Throwable are called **checked** exceptions.

2. (2 pts) If a method has the potential to cause a checked exception to occur, we must either declare that the exception can be thrown or catch and handle the exception.

3. (2 pts) Java supports the **termination** model of exception handling. Given this, will Joanie or Chachi be printed when the following code is executed? Explain.

   ```java
   public class Question3 {
       public static void main(String[] args) {
           int counter;
           for (counter=0;counter<100;counter++)
               if (counter % 3 == 0 && counter % 5 == 0 && counter % 2 == 0)
                   throw new RuntimeException("Multiple of 30");
           if (counter < 30)
               System.out.println("Joanie");
           else
               System.out.println("Chachi");
       }
   }
   ```

   **Neither Joanie nor Chachi will print. Since Java supports the termination model of exception handling, as soon as the exception is thrown, the block that is executing will terminate. That means control never reaches the if.**

4. (2 pts) Point out the two errors in the following code given that IOException is not a subclass of RuntimeException nor of Error. If all of the catch blocks were removed and the program executed, would the message at the top of the stack trace be Hawaii or 50? Explain.

   ```java
   public class Question4 {
       public static void main(String[] args) {
           try {
               int num = Integer.parseInt(args[0]);
               throw new NullPointerException("Hawaii");
           } catch (Exception e) {
               System.out.println("Exception");
           } catch (NumberFormatException nfe) {
               System.out.println("NumberFormatException");
           } catch (java.io.IOException ie) {
               System.out.println("IOException");
           } finally {
               throw new NullPointerException("50");
           }
       }
   }
   ```
The two errors are:

1. The catch block for Exception is listed before its subclasses.

2. There is no potential for a java.io.IOException to be thrown in the try block, and since IOException is not a subclass of RuntimeException nor of Error, it is a checked exception, and therefore we can’t have a catch block for IOException.

If all of the catch blocks were removed, 50 would be printed at the top of the stack trace when the program is executed since the abrupt termination in the finally block overrides the abrupt termination in the try block.

5. (2 pts) The char data type is the only unsigned numeric type since it can’t hold a negative number, and data that is stored in external files is called persistent since it is normally not destroyed when a program ends.