```
/*********************************************************
 ****    Questions 1 - 5 refer to the following code *****
 *********************************************************
 *******************   2 points each  ********************
 *********************************************************/

      int[][]  A  = {
                          {  1,  0, 12, -1 },
                          {  7, -3,  2,  5 },
                          { -5, -2,  2, -9 }
                    };
```

Q1: How many dimensions are in this array?

a) 1
b) 2
c) 3
d) 4
e) 5

Q2: Write a single line of code to change the -3 value in this array to a 5.

Q3: Write a few lines of code to iterate through each value in this array and output the word "found it!" if any given value is the number 2.

Q4: Show how you would create a new array that contained all the same values in this array, plus a 4th sub-array with the values { 4, 22, 9, 0 }. Do not hard-code the values from the previous array into this new array, but rather have them dynamically pulled from the old array and added to the new array.

Q5: Explain how the ArrayList class would make the solution to the previous question easier and quicker to program. Please do not write code.

```
/*******************************************************
 ****    Questions 6 - 12 refer to the following code ****
 *******************************************************
 *******************    5 points each   ********************
 *******************************************************/
```

```
      public class Dog {
            private String name;
            private String furType;
            public boolean hasTail = true;
            private int age = 0;
            private int gender;
            public static final int MALE = 0;
            public static final int FEMALE = 1;
      }
```

Q6: Imagine you are defining a separate class that interacts with this Dog class.... Write one line of code for that class that would instantiate a Dog object from the Dog class and assign a variable called 'fluffy' to point to that Dog object.

Q7: In that same separate class definition, which would be the proper way to refer to the hasTail property of the newly-instantiated Dog?

a) Dog.hasTail
b) fluffy.hasTail
c) hasTail
d) this.hasTail
e) super.hasTail

Q8: Select which instance variables of the Dog class would normally have getters and setters, if their values are to be accessed or modified from code within a separate class definition.

a) name
b) furType
c) hasTail
d) age
e) gender

Q9: Write just the code for a constructor of the Dog class. Using this constructor, it must be possible to instantiate a Dog object that fully represents a 12-year old female dog named Althea, with short fur and a tail.

Q10: Write another overloaded constructor that simply sets the name property of a Dog object.

Q11: Write a getter for the age field.

Q12: Write a setter for the age field that only allows the age field to be set to a value between 0 and 22.

```
/******************************************************
 ****    Questions 13 - 19 refer to the following code ***
 ******************************************************
 ******************    5 points each   *******************
 ******************************************************/


/* The following class definition is in a file named Bannock.java */
package my_recipes

public class Bannock {
	String[] ingredients = {"oat groats", "lard", "water"};
	String cookingMethod = "pan frying";
	boolean leavened = false;
	int cookingTimeQuantity = 5;
	String cookingTimeUnits = "minutes";
	String preparation = "Mix ingredients by hand and form into patties.";
	boolean isCooked = false;
	boolean isMixed = false;
	static String origin = "Scottish";

	public void getRecipe() {
		System.out.println(this.instructions);
	}

	public void mix() {
		this.isMixed = true;
	}

	public void cook() {
		if (this.isMixed) {
			this.isCooked = true;
		}
	}

}
```

Q13: Write the definition of a Scone class that inherits from this Bannock class. This class should override the ingredients, cookingMethod, cookingTime, and leavened instance variables of the Bannock class and assign them new default values that you choose. Do not define any other instance variables or methods in this Scone class.

Q14: Define an instance method of the Scone class named "equals" that compares any given Scone object to another Scone object. This method should accept as its sole parameter a Scone object. The method should return boolean true if the two scones have identical values in the ingredients, cookingMethod, and leavened properties; and it should return boolean false otherwise.

Q15: Write just enough code to show how to instantiate two Scone objects and use the equals() method defined in the previous question to compare the two Scone objects.

Q16: When the equals() method from the previous two questions is called, is the Scone object it requires as an argument passed ...

a) as a value type
b) as a reference type

Q17: Write a method named "setGrain" that goes inside the Scone class that accepts a String as its sole argument. This method should update the value of the first element of the ingredients array for any given Scone object.

Q18: Show how to call any Scone object's mix() method that it inherits from the Bannock class.

Q19: Why might it make conceptual sense to define the "origin" property of the Bannock class as static?

```
/***********************************************************
 *** Questions 20 - 23 share the following answer options   ***
 ***********************************************************
 *********************    2 points each   *********************
 ***********************************************************/

 a) users of a tool should not need to know how that tool works internally
 b) properties and methods "belong to" an object or class
 c) objects can inherit properties from other objects
 d) a class that inherits from a parent class can sometimes be treated as an
     instance of the parent class
```

```
Q20: Encapsulation is the idea that.....
       (indicate one of the answers from the list above)


Q21: Abstraction is the idea that.....
       (indicate one of the answers from the list above)


Q22: Inheritance is the idea that.....
       (indicate one of the answers from the list above)


Q23: Polymorphism is the idea that.....
       (indicate one of the answers from the list above)
```

```
/*****************************************
 *** Questions 24 - 30 are conceptual   ***
 *****************************************
 *************   2 points each  ***********
 *****************************************/
```

Q24: Explain succinctly why the special keyword 'this' is never used in static methods.

Q25: Primitive data types and Strings are value types, not reference types.

a) yes, that's correct
b) no, that's utter rubbish

Q26: There is a special keyword 'parent' in object-oriented programming.

a) yes, that's correct
b) no, that's utter rubbish

Q27: Any class definition can have only one constructor.

a) yes, that's correct
b) no, that's utter rubbish

Q28: Let say, hypothetically-speaking, that the PApplet class contains a method named "draw()". Let's also say that the MyGame class extends the PApplet class. This means that the MyGame class cannot ever define a method of its own named "draw()".

a) yes, that's correct
b) no, that's utter rubbish

Q29: Creating a new Integer object by instantiating the Integer class from the Java API, as in the following code, gives you a new primitive data value stored in the variable x.

```
             Integer x = new Integer(5);
```

a) yes, that's correct
b) no, that's utter rubbish

Q30: The special 'super' keyword points to the class from which a given object inherits.

a) yes, that's correct
b) no, that's utter rubbish

```
/*****************************************
 ***********   EXTRA CREDIT!!!   **********
 *****************************************
 **************   5 points  **************
 *****************************************/
```

Clearly explain two benefits of doing coding in an object-oriented style.

This is extra paper

This is extra paper

This is extra paper

This is extra paper