Question 1. [5 points] Consider the following code:

```java
int a = 9;
int b = 4;
int c = a [op] b;
System.out.printf("%d\n", c);
```

If the code prints the output 1, what is [op]? (Note that the %d placeholder prints an integer.)

Question 2. [5 points] Consider the following class (which begins on the left and continues on the right):

```java
public class Letter {
    public char c;
    public Letter(char cVal) {
        c = cVal;
    }
    public static void main(String[] args) {
        Letter l1 = new Letter('X');
        Letter l2 = new Letter('Y');
        l2 = l1;
        l1.c = 'Z';
        System.out.printf("%c\n", l2.c);
    }
}
```

What output is printed by the main method of the class? (Note that the %c placeholder prints a character.)

Question 3. [5 points] What output is printed by the following statements (which begin on the left and continue on the right)?

```java
int[] a = new int[1];
int[] b = new int[1];
a[0] = 23;
b[0] = a[0];
```

```java
public static void main(String[] args) {
    a[0] = 49;
    System.out.printf("%d\n", b[0]);
}
```
**Question 4.** [5 points] Consider the following class (which begins on the left and continues on the right):

```java
public class Q4 {
    public static void f(int x) {
        x = 42;
    }

    public static void main(String[] args) {
        int y = 39;
        f(y);
        System.out.printf("%d
", y);
    }
}
```

What output is printed by the `main` method of this class?

**Question 5.** [20 points] Consider the following JUnit test class (which begins on the left and continues on the right):

```java
public class TeamTest {
    private Team team1;
    private Team team2;

    @Before
    public void setUp() {
        team1 = new Team("Spartans");
        team2 = new Team("Trojans");
    }

    @Test
    public void testGetName() {
        assertEquals("Spartans",
                team1.getName());
        assertEquals("Trojans",
                        team2.getName());
    }

    @Test
    public void testGetInitialWins() {
        assertEquals(0, team1.getWins());
    }

    @Test
    public void testGetInitialLosses() {
        assertEquals(0, team1.getLosses());
    }

    @Test
    public void testIncrementWins() {
        team1.incrementWins();
        assertEquals(1, team1.getWins());
        team1.incrementWins();
        assertEquals(2, team1.getWins());
        // team2 shouldn’t have changed
        assertEquals(0, team2.getWins());
    }

    @Test
    public void testIncrementLosses() {
        team2.incrementLosses();
        assertEquals(1, team2.getLosses());
        team2.incrementLosses();
        assertEquals(2, team2.getLosses());
        // team1 shouldn’t have changed
        assertEquals(0, team1.getLosses());
    }
}
```

[Question continued on next page.]
An instance of the Team class has a name, number of wins, and number of losses. A Team object’s name is set by the constructor when the object is created. The object’s numbers of wins and losses start at 0, but can be increased later by calling the incrementWins and incrementLosses methods, respectively.

Define the Team class so that all of the tests in the TeamTest class succeed. Note that the setUp method will be called before each test method, so each test method gets fresh team1 and team2 objects.

Don’t forget to define fields and a constructor.
Question 6. [10 points] Using the Team class you wrote in the previous question, complete the playGame method below. If the coinFlip parameter is an odd value, then teamA's number of wins should increase by 1 and teamB's number of losses should increase by 1. If the coinFlip parameter is an even value, then teamA loses and teamB wins.

Example JUnit test:

```java
Team team1 = new Team("Dimwell");
Team team2 = new Team("Dolly Sisters");

playGame(1, team1, team2);
assertEquals(1, team1.getWins());
assertEquals(0, team1.getLosses());
assertEquals(0, team2.getWins());
assertEquals(1, team2.getLosses());

playGame(2, team1, team2);
assertEquals(1, team1.getWins());
assertEquals(1, team1.getLosses());
assertEquals(1, team2.getWins());
assertEquals(1, team2.getLosses());
```

```java
public static void playGame(int coinFlip, Team teamA, Team teamB) {
```
Question 7. [10 points] Consider the following interface:

```java
public interface ArrayComputation {
    public double compute(double[] arr);
}
```

Define classes called `ArraySum` and `ArrayProduct` which implement this interface. `ArraySum`'s `compute` method should return the sum of the elements of its argument array, while `ArrayProduct`'s `compute` method should return the product of the elements of its argument array.

Example JUnit tests:

```java
double delta = 0.000001;
double[] a = new double[3];
a[0] = 8.3;
a[1] = 3.4;
a[2] = 1.4;

ArrayComputation sum = new ArraySum();
ArrayComputation product = new ArrayProduct();

assertEquals(8.3 + 3.4 + 1.4, sum.compute(a), delta);
assertEquals(8.3 * 3.4 * 1.4, product.compute(a), delta);
```
Question 8. [10 points] Complete the `removeEvenLengthStrings` method below. It should remove all of the even-length strings from the `ArrayList` given as a parameter. You can call the `length` method on a string to determine its length.

Example JUnit test:

```java
ArrayList<String> nonsense = new ArrayList<String>();
nonsense.add("colorless"); // length 9
nonsense.add("purple"); // length 6
nonsense.add("sheep"); // length 5
nonsense.add("seethe"); // length 6
nonsense.add("starkly"); // length 7

removeEvenLengthStrings(nonsense);
assertEquals(3, nonsense.size());
assertEquals("colorless", nonsense.get(0));
assertEquals("sheep", nonsense.get(1));
assertEquals("starkly", nonsense.get(2));
```

```java
public static void removeEvenLengthStrings(ArrayList<String> list) {

```
Question 9. [30 points] Programming problem — see instructions in class.