Circle the bullet in front of each statement that accurately describes your instructor's course policies:

- The final exam for this course will be comprehensive.
- Lab attendance is required.
- You are expected to read the lab assignment before lab begins.
- If you miss more than three lectures, your grade will be dropped by half a letter grade.
- You must submit every lab assignment in order to pass this course.
- Homework is typically due before taking a weekly quiz.
- All students must submit a copy of their lecture notes before the end of the day.
- Having a copy of all or part of another student's source code, just to look at for a little help, is considered cheating.
- Your instructor is such a nice guy that if you turn in an assignment 45 minutes late, he wouldn't consider it late.
- Your instructor will give you a free lunch if you invite him to lunch.

Do not circle the bullet in front of any statement that incorrectly describes your instructor's course policies.

Feel free to provide an explanation for your answer if you believe the statement to be ambiguous.

Implement the add method found in the IntegerHolder class below. Your implementation should be very similar to the add method implementation done in lecture yesterday. The main difference is that the value passed to the method should be added in the location specified by the index argument. For example,

```
ih.add(0, 5);          // adds a 5 to the beginning of the data array
ih.add(ih.size(), 5); // adds a 5 to the end of the data array
```

If the index value is less than zero or greater than the size of the IntegerHolder object, the method should throw an IndexOutOfBoundsException.

```
public class IntegerHolder {
  private int size;
  private int[] data;
  private static final int INITIAL_CAPACITY = 10;
  private static final double GROWTH_RATE = 1.5;

  // Constructors and add(int value) omitted to save space

  public boolean add(int index, int value) {









    return true;
  }
}
```

Implement the removeFirst method for the Linkest class discussed in lecture this week. The Linkest class had the following attributes: size, head, and tail. The nested Node class has the following attributes: value (a String) and next. The removeFirst method should return a reference to the String being removed from the list. If the list is empty, the method should return `null`.

```
public class Linkest {

  // ...

  public String removeFirst() {









  }
}
```

Give the asymoptotic time complexity for each of the following operations. Be sure to justify your answer with an explanation.

(a) `list.add(0, Math.PI);` where `list` is an `ArrayList<Double>` object and `n = list.size();`.

(b) `list.add(0, Math.PI);` where `list` is a `LinkedList<Double>` object and `n = list.size();`.

(c) `list.add(n/4, Math.PI);` where `list` is an `ArrayList<Double>` object and `n = list.size();`.

(d) `list.add(n/4, Math.PI);` where `list` is a `LinkedList<Double>` object and `n = list.size();`.

In the lab 4 assignment, you were required to implement the LinkedRosterIterator inner class. Show the attributes required for the class, and implement the constructor and hasNext() method. You may assume that the outer class has the following attributes: head and tail (both references to Entry objects) and an integer, size.

```
private class LinkedRosterIterator implements Iterator<Student> {
  // Declare attributes here









  private LinkedRosterIterator() {









  }

  private boolean hasNext() {









  }
}
```

Every integer is evenly divisible by 2^i where $0 \leq i < \infty$, i.e., every integer can be evenly divided by 2 zero or more times. We can determine the value of i by seeing how many times we can divide the integer by 2 and still get an integer result.
Complete the recursive method, quiz6, that returns the value of i.

```
// Does not need to work for n<=0
public static int quiz6(int n) {
  int answer = 0;
```

```
  return answer;
}
```

In order to provide you with the best possible educational experience, I would like to get your comments on what is and isn't working in this class and lab.

At what moment were you most involved (excited, enthusiastic, . . .) in class/lab?

At what moment were you least involved (bored, disconnected, . . .) in class/lab?

What was the most helpful action taken by anyone in class/lab?

What was the most confusing action taken by anyone in class/lab?

What most surprised you?

What would be the first thing you would do differently if you were teaching the class?

Please add any additional comments you have. Indicate if you do *not* wish to have them appear in the summary returned to the class. Use the back of this page if necessary.

Implement the recursive height(Node<E> node) method following the height definition used in class. The non recursive version of the height method is shown below. It should return -1 if the tree is empty.

```
public int height() {
  return height(root);
}
```

Consider the Word class shown below:

```
public class Word implements Comparable<Word> {
  private String word;

  public Word(String word) {
    this.word = word;
  }

  public String toString() {
    return word.toString();
  }

  public boolean equals(Object obj) {
    return (compareTo((Word) obj) == 0);
  }

  public void setWord(String word) {
    this.word = word;
  }

  public int compareTo(Word arg0) {
    return word.compareTo((arg0).getWord());
  }

  public int hashCode() {
    return word.hashCode();
  }
}
```

Show how the Word class must be modified so that objects from this class would be inserted into a HashSet<Word> using the length of the String contained in the word field instead of the default technique for determining the appropriate bucket.