AP Computer Science Summer Packet.

I want to hit the ground running next year. There's a lot of cool things we can do **IF** everyone already knows the basics.

These problems should not be hard. They are all from around the time of your midterm in intro, but it's important that you be able to do these with little to no thought. They should be second nature to you. All together they took me about 2 hours of work. If they are taking you more than about 3, that's a sign that you need to do a lot more practice, or that you should wait a year before taking AP cs.

If you have any questions or concerns in that regard, please contact me at david.dobervich@gmail.com, which I will be checking all summer.

Do you want to do more interesting problem? If you think it will be a waste of time for you to do these problems, contact me. I will have you come argue your case. You may do a shorter, more interesting (ie difficult) set of problem instead if you wish, or a short project of your own choosing. You may not, however, do nothing; you will need to do some problem set.

Turn-in Instructions

Problem Set	**Due Date** (before midnight on the given date)
1	06/22/14
2	06/29/14
3	07/06/14
4	07/13/14
5	07/20/14

How to turn in

Email: david.dobervich@gmail.com

Subject line: “Your name, Summer Problems Set N”

Attach: A single document (could be .java, .txt, .doc, .pdf) with the solutions **as well as any questions you have** or anything you'd like to let me know about this problem set.

The Questions

These materials are drawn from two excellent online books:

(1) Think Java: http://www.greenteapress.com/thinkapjava/html/index.html

(2) Princeton's Intro Java Course: http://introcs.cs.princeton.edu/java/home/

I will also be posting whirlwind review videos on my youtube channel covering the different problem sets for your viewing pleasure.

Problem Set I: Input, output, data types, and standard math operations.

1. Some of these declarations and assignments have mistakes. Identify them. (Be as specific as you can about what's wrong and/or how to fix it). (Assume these lines are all part of one program; so it might declare some variables and then assign values later).

```
1 int a, b, c;

2 String number = 9;

3 int = 9;

4 double number = 9;

5 boolean done = true;

6 String number = “9”;

7 String name;

8 name = “wallaby”;

9 7 = a;

10 int num = 3.14;
```

For each of the 10 lines above, if there is a problem, describe what it is and how to fix it. If it's ok, write "ok".

(2) Say what data type would be the best to use to store each of the following kinds of information and why. If there is more than one answer that could work, list them all and then tell me which you would choose and why. If you want to use more than one variable to represent this information, explain.

(a) The price of a plane ticket

(b) How many pets you own

(c) Your mother's maiden name

(d) Whether or not a user has paid their registration fee

(e) A product's id number in a warehouse

(f) A date for a calendar

(g) Whether or not a game is over yet

(h) The location of a queen on a chess board

(i) If an animal is alive or dead

(3) Consider the following code:

```
a = a + b;

if (a < 0) {

    c = c * (-1);

    b = 5;

}

a = a + b;

if (a < 0) {

    c = c * (-1);

}
```

(a) If a = 9, b = -7, and c = 25, before the code starts, what will the values of a, b and c be when the code is finished running? (you should be able to answer these without actually running the program).

(b) If a = 9, b = -11, and c = 25, before the code starts, what will the values of a, b and c be when the code is finished running?

(4). Here is a simple program which converts miles to feet.

```
public class Example {

    public static void main(String[] args) {

        double miles = 3.1;

        double feet = miles / 5280;

        System.out.println("There are " + feet + " feet in " + miles + " miles");

    }

}
```

a. Run the program. Experiment changing the value assigned to miles. (you don't have to answer any questions here).

b. Write a program which declares a double variable called radius. Calculate the circumference and area of a circle with that radius and display the results.

(5). ints and doubles have maximum and minimum values they can hold. There are special, pre-defined constants called MAX_VALUE and MIN_VALUE which contain these maximum and minimum values.

(a) Run the following program and tell me what max and min are for your system (they might vary depending on what computer you run java on).

```
public class Example {
    public static void main(String[] args) {
        int max = Integer.MAX_VALUE;
        int min = Integer.MIN_VALUE;
        double maxDouble = Double.MAX_VALUE;
        double minDouble = Double.MIN_VALUE;

        System.out.println("Maximum integer is: " + max);
        System.out.println("Minimum integer is: " + min);
        System.out.println("Maximum double is: " + maxDouble);
        System.out.println("Minimum double is: " + minDouble);
    }
}
```

(b) Create a program with a variable *a* declared as `int a = Integer.MAX_VALUE`. What do each of the following print? Explain each outcome as best you can:

i. System.out.println(a + 1);

ii. System.out.println(2 - a);

iii. System.out.println(-2 - a);

iv. System.out.println(2 * a);

v. System.out.println(4 * a);

(6). The following two programs are exactly the same as problem 4, but they illustrate two different ways you might prompt the user for input. Try each program. Make sure you know how they work.

```
import java.util.Scanner;

public class HelloWorld {
  public static void main(String[] args) {
    Scanner userInput = new Scanner(System.in);
    double miles = userInput.nextDouble();

    double feet = miles / 5280;
    System.out.println("There are " + feet + " feet in " + miles + " miles");
  }
}
```

```
import javax.swing.JOptionPane;

public class HelloWorld {
  public static void main(String[] args) {
    String response = JOptionPane.showInputDialog("Enter a number of miles");
    double miles = Double.parseDouble(response);

    double feet = miles / 5280;
    System.out.println("There are " + feet + " feet in " + miles + " miles");
  }
}
```

Modify your program from part 5b to prompt the user for the radius before calculating the area and circumference.

(7). Many common math operations are available using the Math library. You use them by writing

```
output = Math.yourFunction(input);
```

For example, the square root function can be used this way:

```
output = Math.sqrt(2);
```

Here is a simple code excerpt that calculates the distance between the points (x1, y1) and (x2, y2) using the distance formula.

```
double x1 = 3, y1 = 5;
double x2 = 1, y2 = 11;
double dx = x2 - x1; // difference b/t x values
double dy = y2 - y1; // difference b/t y values
double distance = Math.sqrt(dx*dx + dy*dy);
```

Note that to square a number, I cannot say a^2; I must say a*a.

a. **Area of a triangle.** Write a program that prompts the user for three doubles: a, b, and c, representing the side lengths of a triangle. It should calculate the area of the triangle using Heron's formula:

$$area = \sqrt{s \cdot (s-a) \cdot (s-b) \cdot (s-c)}$$

where $s = \frac{a+b+c}{2}$

(8). There are many other important functions in the Math library. Some particular ones to note:

```
Math.max
Math.min
Math.pow
Math.abs
Math.rand
```

Find the lines which use these methods in the following example program which illustrate many of the Math library methods: http://www.cafeaulait.org/course/week4/40.html

For a more explicit tutorial on their use, you can consult this page: http://www.functionx.com/java/Lesson17.htm

Wind chill. Given the temperature t (in Fahrenheit) and the wind speed v (in miles per hour), the National Weather Service defines the effective temperature (the wind chill) to be:

$$w = 35.74 + 0.6215\ t + (0.4275\ t - 35.75)\ v^{0.16}$$

Write a program that prompts the user for the two doubles, t and v and prints out the wind chill. Use Math.pow(a, b) to compute a^b. Note: the formula is not valid if t is larger than 50 in absolute value or if v is larger than 120 or less than 3 (you may assume that the values you get are in that range).

(10). Math.rand() returns a double which is greater than or equal to 0, and less than 1.

A very important concept you will be using a lot is an idea called **casting**. Casting means "converting a value in to another data type". The way you cast is by writing the data type you wish to convert to in parenthesis before the value you wish to convert. For example:

```
int n = (int)5.6; // this converts 5.6 into an integer by truncating (ignoring) the decimal part.
```

a. What is the largest and smallest possible value for each of these variables?

```
int a = (int)(Math.rand()*3);

int b = (int)(1 + Math.rand()*3);

int c = (int)(-10 + Math.rand()*5);

int d = (int)(1 + Math.rand()*100);
```

b. Write a program that will simulate rolling two 6-sided die using Math.rand(). Display the value of each die and their sum. For example, a sample output from your program might be:

```
"Die 1: 4 Die 2: 1 Sum: 5"
```

Set II: If-statements and loops

1. The following if-statement will give a compile-time error:

```
if (10 < x < 20)
    System.out.println("x is between 10 and 20");
```

You are really performing two tests: `10 < x` and `x < 20`. In java you must separate these tests explicitly.

```
if ((10 < x) && (x < 20))
    System.out.println("x is between 10 and 20");
```

The x && y is only true if both x and y are true. If you want to test whether *at least one* of two things is true, you can use: x || y. For example, to test whether x is outside the range 10 to 20, we could write:

```
if ((10 > x) || (x > 20)) System.out.println("x is not between 10 and 20");
```

Notice the two tests in the if-statement. We are testing if x is smaller than 10, or if x is larger than 20: the two ways x could be outside the interval we want. If *either* one of these is true, x is definitely outside the interval.

a. Write a program which generates a random number between 1 and 5, and another one between 5 and 10. It should then prompt the user to enter an integer. Display all three numbers. If the user's number lies between the two random numbers, display the word "pie".

2. Consider the following code:

```
System.out.println(“A”);
if (b < c) {
  System.out.println(“B”);
  if (a != c) {
    System.out.println(“C”);
  } else {
    System.out.println(“D”);
  }
  System.out.println(“E”);
} else if (b < a) {
  System.out.println(“F”);
  if (a > c) {
    System.out.println(“G”);
  } else if (a == c) {
    System.out.println(“H”);
  } else {
    System.out.println(“I”);
  }
  System.out.println(“J”);
} else {
  System.out.println(“K”);
}
```

Say what the above code will display for each of the following sets of values for a, b, and c.

```
(A) a = 1, b = 2, c = 3
(B) a = 1, b = 3, c = 2
(C) a = 2, b = 1, c = 3
(D) a = 2, b = 1, c = 1
(E) a = 1, b = 1, c = 1
(F) a = 3, b = 3, c = 2
(G) a = 3, b = 2, c = 1
```

3. Here is a short program that generates two numbers between 1 and 30.

```
int a = (int)(Math.rand()*30 + 1);
int b = (int)(Math.rand()*30 + 1);
```

a). Write an if-statement that will display which number is larger. (assume they're not the same).

b). Write an if-statement that will display “true” if a and b are within 15 numbers of each other.

c). Write an if-statement that will display “true” if a is between 0 and 10 AND b is between 90 and 100

4. Write a program to prompt the user to enter 2 integers. Write a for-loop which will loop between the two integers and display every number in between. note: you cannot assume they will enter the smaller number first. **Your program should work no matter what order they enter the two numbers**. If their numbers are the same, it should just output their number. Here are some sample outputs:

sample output 1	*sample output 2*	*sample output 3*

`Enter a number: 16`
 	
`Enter another number: 4`
 	
`4 5 6 7 8 9 10 11 12 13 14 15 16`
	`Enter a number: 6`
	
	`Enter another number: 6`
	
	`6`
		`Enter a number: -3`
		
		`Enter another number: 2`
		
		`-3 -2 -1 0 1 2`

5. This program will be a combination of ideas from 1a and 2. Write a program which generates two random numbers between 1 and 100. Use a for-loop to let the user guess an integer 5 times. If their integer lies between the two random numbers, display a message that they won a round. At the end of the loop, tell them how many rounds the won. [hint: use an if-statement or Math.min to figure out which of the two random numbers is the smaller one before you use them in your if-statement].

6. Consider the following loop:

```
for ( int i = 0; i < 10; i = i + 1) {
    System.out.print( i + " ");
}
```

Variation #1: Displays 2 4 6 8 ... 18	Variation #2: ALSO displays 2 4 6 8 ... 18

`for ( int i = 2; i <= 18; i = i + 2) {`
`  System.out.print( i + " ");`
`}`
	`for ( int i = 1; i < 10; i = i + 1) {`
	`  System.out.print( 2*i + " ");`
	`}`

Notice that you when create a for-loop, you can either directly control the values of the loop variable (as in variation #1) or leave them unchanged and perform a calculation with the loop value to get the value you desire (as in variation #2).

a. How would you modify the for-loop (NOT the print statement) so it will display...

```
112 113 114 115 116 ... 203
```

b. How would you modify the for loop so it will display:

```
2 4 6 8 10
```

c. How would you modify the for-loop so it will display...

```
10 9 8 7 6 5 4 3 2 1
```

d. Leave the for-loop unchanged from the original eample. How could you change the print statement so it displays:

```
3 5 7 9 11 13 15 17 19 21
```

7. Each of these code excerpts can be re-written in a better (shorter, more direct) way. Please do so.

a. Sometimes students leave an empty block like the example below. That's (barely) ok for an intro class; we should do better. Write a fixed version of each of these code samples.

```
if (n > 0) {
  // do nothing
} else {
  doSomethingWith(n);
}
```

```
if ( runTest() == true ) {
  // do nothing
} else {
  doSomethingWith(n);
}
```

b. There is duplicated code in the excerpt below. It could be made shorter by moving some of this code outside of the if-statements. Do this.

```
if (x > 0) {
  method1();
  z = method2(x);
  x = -x;
} else {
  method1();
  z = method2(x);
  x++;
}
```

c. Use a loop to make this series of statements shorter. (You should loop 9 times and only have 1 statement inside your loop).

```
a[0] = 30;
a[1] = 40;
a[2] = 50;
a[3] = 60;
a[4] = 70;
a[5] = 80;
a[6] = 90;
a[7] = 100;
a[8] = 110;
```

8. Here is a program that will display a table of values for the function $f(x) = x^2$.

```
for (int i = 0; i < 10; i++)
    System.out.println("f(" + i + ") = " + i*i);
```

Consider the function $f(x) = 2x^3 - x + 1$. Use a for-loop to display a table of values for this function from x = 0 to x = 20. Here is a sample output:

```
x = 0 and f(0) = 1
x = 1 and f(1) = 2
x = 2 and f(2) = 15
....
x = 20 and f(20) = 15981
```

9. Consider the following program:

```
for (int j = 0; j < i; j++) {
    System.out.print(j + " ");
}
```

This will display the numbers from 0 to i (not including i). Now consider what happens if we place this code inside a loop that changes the value of i.

```
for (int i = 0; i < 5; i++) {
    for (int j = 0; j < i; j++) {
        System.out.print(j + " ");
    }
}
```

a. What will this display?

b. Modify it so that each line counts down to 0 (instead of up from 0).

c. Modify it again (by adding another loop) so that when it gets to 0, it keeps displaying 0 so the output is shaped like a square. For example, it should display.

```
0 0 0 0 0

1 0 0 0 0

2 1 0 0 0

3 2 1 0 0

4 3 2 1 0
```

10. Write a program which prompts the user to enter a positive integer. Display a triangle of "*" characters starting with whatever number they typed. For example, if they had entered 6, it would display:

```
* * * * * *

* * * * *

* * * *

* * *

* *

*
```

[hint: use a for-loop to count down from the user's number. Inside the for-loop, make another for-loop which will display that many * star characters. Use System.out.print instead of System.out.println].

11. Write a program using a loop and four if statements to print:

```
12 midnight

1am

2am

...

11am

12 noon

1pm

...

11pm
```

*12. Write a program that prompts the user for a positive integer *n* and prints a (2N + 1)-by-(2N + 1) "x"-shape like the one below. Use two for loops and one if-else statement.

```
* . . . . . *

. * . . . * .

. . * . * . .

. . . * . . .

. . * . * . .

. * . . . * .

* . . . . . *
```

Set III: Methods

0. Know what the following terms mean: method signature, parameters, arguments, return type, return value, method name.

1. Write the following method signatures...

a. A method called distance that takes 4 ints as parameters and returns a double.

b. A method called isOverlapping which takes 4 ints as parameters and returns a boolean.

c. A method called display which takes no parameters and has no return value.

d. A method called toString which takes no parameters and returns a String.

2. There are several problems with the following code. Many of the inputs to the methods don't match the method signature, or the return value is being assigned to the wrong data type. Identify all the problems having to do with the inputs and outputs of the 4 methods below.

```
public static void main(String[] args) {
  String a, b;
  int c = 1, d;
  double e = 0.1, f = 0.2;

  c = mystery1();
  e = mystery2(e, a, f);
  a = mystery3(1, 2, e);

  if ( mystery4("bird", 10 ){
    System.out.println("mystery4 has done it again!");
  }
}

public static double mystery1(int a)
  // code for mystery 1
}

public static double mystery2(double a, double b, double c)
  // code for mystery 2
}

public static String mystery3(double a, double b, double c)
  // code for mystery 3
}

public static void mystery4(String a, int b)
  // code for mystery 4
}
```

3. What will the following code display?

```
public static void main(String[] args) {
  int a = 10, b = 20, c, d;
  c = modify(a, b);
  d = modify( pump(a), b);

  System.out.println(c + " " + d);
}

public static int modify(int x, int y) {
  if (x > y) {
    return -x;
  } else {
    return -y;
  }
}

public static int pump(int p) {
  return p*p - 10;
}
```

4. Write methods to perform the following tasks

a. Write a method called max which takes 3 ints as parameters and returns the largest of the 3. [hint: Use Math.max() several times].

b. Write a method called average which takes two ints as parameters and returns a double representing their average.

> Note: In java: double a = 2 / 3; will assign 0 to a because 2 and 3 are both int values it performs an integer division which truncates any decimal portion of the result. To fix this problem, make one or both of the values in the division a double. For example:

Solution #1: double a = 2.0 / 3; OR double a = (double)2/3;

c. Write a method called displayNTimes which takes a String and an int as parameters and displays the string the number of times specified by the int.

For example:

> displayNTimes("bird", 3); // this should display "birdbirdbird"

d. Write a method called repeatNTimes which works exactly like method c, except that instead of displaying the result, it should return a String which is the input word repeated that many times.

For example:

> String a = repeatNTimes("yo", 4); // a is now equal to "yoyoyoyo"

[hint: if b is a string, then b = b + "hi"; will append "hi" to the end of b. Do this inside your loop instead of displaying].

5. Here is a table used to determine what type of category a hurricane is according to its wind speed (called the Saffir-Simpson scale). If the wind speed is less than 74mph it does not qualify as a hurricane.

Category	Wind Speed (mph)
1	74 - 95
2	96 - 110
3	111 - 130
4	131 - 155
5	155 and above

Write a method called hurricaneType which takes an int representing wind speed as a parameter, and returns an int representing the category of the hurricane whose wind-speed is given by the input. Use a return-value of 0 to denote that the input is not a hurricane.

Set IV: Simple classes

1. Imagine you have a Dog class, and a Dog object named fido.

Fido has two important methods for you:

fido.run() will make fido run for a short distance. (It doesn't return anything).

fido.isTired() will return **true** if fido is tired, **false** if fido isn't tired yet.

Write a loop that will make fido run until he gets tired. You should put a variable in your loop to count how many times you had to tell fido to run until he got tired.

At the end of your program, you should display a message such as:

"Fido got tired after running 11 times"

2. This problem is totally different from the previous one. No connection! But it's also about a dog.

Create a class to represent a Dog.

Your dog is a virtual dog, so you only need to remember a few pieces of information about it. Use the test-program below to figure out

1). What information (fields) your Dog class should have, and

2). What methods your Dog class should have.

```
public class Tester {
  public static void main(String[] args) {
    Dog fido = new Dog("fido", 23);
    Dog rex = new Dog("rex");

    System.out.println("Fido's age is " + fido.getAge() ); // displays 'Fido's age is 23'
    System.out.println("Rex's age is " + rex.getAge() ); // displays 'Rex's age is 0'

    fido.sit();

    if ( fido.isSitting() == true ) {
      System.out.println("Fido is sitting!);
    }

    fido.stand();

    if ( fido.isSitting() == false) {
      System.out.println("Fido is NOT sitting!");
    }

    System.out.println(rex); // displays: 'name: rex, age: 0'
  }
}
```

a). Write out the methods you need for the above to work (including constructors)

Return value	Method name	Arguments
--	Dog	
--	Dog	
String	toString	--

b). What pieces of information (fields) do you need to have?

c). Implement (create) the class.

3. Create a new class to represent a sphere. Your class should have the following methods:

A constructor that takes a radius as an input.

A method called *getRadius* which will return the radius of the sphere.

A method called *setRadius* which will set the radius of the sphere.

A method called *volume* that should *return* the volume of the sphere. (Don't display it in this method!). The volume of a sphere can be calculated this way: volume = (4/3) * 3.14159 * r^3

A *toString* method which should return a string something like the following:

"Your circle has radius: 5"

a). implement (create) the class.

b). implement (create) a test-class that will test your sphere class. You should be sure to test each of the methods you've made.

Set V. Arrays and ArrayLists

In Java, the two main List data types are arrays and ArrayLists. Here's a quick review of how to do the basic operations with each.

Arrays	**ArrayList**

Creating
 	
arrays can hold either primitive data types or objects.
 	
`int[] intList = new int[100];`
`String[] names = new String[5000];`
`boolean[] flags = new boolean[100];`
`Person[] people = new Person[10];`
	Creating
	
	ArrayLists can **only** hold objects.
	
	`ArrayList<Integer> intList = new ArrayList<Integer>();`
	`ArrayList<String> names = new ArrayList<String>();`
	`ArrayList<Person> people = new ArrayList<Person>();`
	`ArrayList<Object> stuff = new ArrayList<Object>();`

Read a value from the list
 	
`int first = intList[0];`
`System.out.println( names[10] );`
	Read a value from the list
	
	int first = intList.get(5);
	System.out.println(names.get(10));

Write a value to the list
 	
`intList[0] = 10;`
`people[5] = new Person("waldo");`
	Write a value to the list
	
	intList.add(10); // add 10 to the end of the list
	
	names.add(0, "bingo"); // insert an element at index 0. All other elements // shift down
	
	names.set(5, "waldo"); // set element 5 to "waldo". Old element is // overwritten

Get the size of the list System.out.print(intList.length);	**Get the size of the list** System.out.println(intList.size());

Loop over a list (for loop)
 	
for (int i = 0; i < people**.length**; i++) {
  System.out.println(**people[i]**);
}
	Loop over a list (for loop)
	
	for (int i = 0; i < people**.size();** i++) {
	  System.out.println(**people.get(i)**);
	}

Loop over a list (for-each loop)
 	
for (String name : names) {
  System.out.println(name);
}
	Loop over a list (for-each loop)
	
	for (Person p : people) {
	  System.out.println(p);
	}

0. Write code which creates an int array with 1000 elements. Loop through it and fill it with randomly generated ints between 1 and 10 (inclusive).

1. Complete the code below. You should (a) create a String array to hold 100 elements, (b) loop through the array and assign each element a String consisting of a randomly chosen name from the names array, the word " the " and a randomly chosen adjective. For example, "behemoth the fallow", "doctorow the somewhat large" and so on.

```
String[] names = ["abadaba", "behemoth", "carlheinz", "doctorow"];
String[] adjs = ["the fallow", "the somewhat large", "the incredibly hungry", "the bemused"];
                                                        // create a 100 element String array called phrases
for( ____________________________) {                    // loop through it
                                                        // get a random element from names
                                                        // get a random element from adjs
                                                        // concatenate them together
                                                        // assign them to the next element of your array
}
```

* * *

Here are three code excerpts that show how to do common things with arrays. They may be helpful in answering the questions below.

Find the minimum value. (or maximum value).

```
int minSoFar = array[0];
for(int i = 0; i < array.length; i++)
      if (array[i] < minSoFar)
            minSoFar = array[i];
```

Swap two elements. This will swap the elements in index i and j.

```
int temp;
temp = array[i];
array[i] = array[j];
array[j] = temp;
```

Copy the array.

```
int[] newCopy = new int[ array.length ];
for (int i = 0; i < array.length; i++)
      newCopy[i] = array[i];
```

2. The following code is supposed to find the index of the largest value in an array.

```
int largestIndex = 0;
for (int i = 1; i < array.length; i++) {
    if (array[i] > array[i-1]) {
        largestIndex = i;
    }
}
```

a). Invent an example of an array where this code will fail to find the index of the largest value.
b). Explain in English what the problem with the code is.
c). Write a version which works.

3. Write a method shuffle(int[] a) which will shuffle (randomize the order of) the values in a. [hint: loop some large number of times. Select two random indexes from the array. Swap the values at those two indexes].

4. Given an array of ints called *nums*, return true if the array contains no 1's and no 3's.

```
public boolean contains1or3(int[] nums) { ... }
```

5. Write a method which returns true if its two input arrays are equal. (Two arrays are equal if they have the same size, and the same values in the same order; note: you will need to loop through the arrays to check each value. Make sure you don't go out of bounds!).

6. Write code which creates an ArrayList to hold Integers. Loop through it and fill it with randomly generated Integers between 1 and 10 (inclusive). There should be 1000 random numbers in total.

7. **Insert Leading Silence in an Audio Sample**. It is possible to store sound digitally as an ArrayList of integer values corresponding to the heights of the sound wave sampled at regular intervals. A common operation someone may wish to do is to add an interval of silence at the beginning of the audio sample. In this problem you will write a method called addLeadingSilence which takes as parameters an integer ArrayList, sample, representing the audio sample, and an integer length representing the length of silence to be added at the beginning of the sample. You will return a new integer ArrayList whose length is the sum of the original sample ArrayList and the integer length. The first length elements of the ArrayList will contain the value 0 (representing silence), and the remainder of the ArrayList will be an exact copy of sample.

For example, assume that the sample ArrayList has been initialized to: {1034, 1036, -2394, 955, 3, 45, 22};

```
ArrayList<Integer> result = addLeadingSilence(sample, 4);

result equals: {0, 0, 0, 0, 1034, 1036, -2394, 955, 3, 45, 22};
```

Complete method addLeadingSilence below.

```
/** Return a copy of sample with length leading 0.
* The total length of the return ArrayList is sample.length + length
* @param sample the audio sample
* @param length the number of leading 0's to add
* Preconditions: sample contains at least one value. length > 0.
* @return a new integer ArrayList with length leading 0's, followed by an exact
* copy of sample.
*/
public static ArrayList<Integer> addLeadingSilence(ArrayList<Integer> sample, int length)
```

(8) Extracting an Audio Sample

It is possible to store sound digitally as an ArrayList of integer values corresponding to the heights of the sound wave sampled at regular intervals. A common operation someone may wish to do is to extract a sample from an audio clip. In this problem you will write a method called extractSample. It will return an ArrayList that's an exact copy of sample starting at index start and ending at index end.

For example, assume that the samples ArrayList has been initialized to the following values:

40	345	-358	83	67	0	29	-447	21	199	-456	-332	2	98	75	65

When the statement:

```
ArrayList<Integer> result = extractSample(sample, 2, 7);
```

is executed, result will be:

```
-358 83 67 0 29 -447
```

Complete method extractSample below.

```
/** Extracts an ArrayList of values from sample between indexes start and end.
* @param sample the integer ArrayList representing the audio sample.
* @param start the ArrayList index where the extraction should start.
* @param end the ArrayList index where the extraction should end.
* Preconditions: sample contains at least one value.
* 0 <= start < end < samples.length
* @return a new integer ArrayList containing the values of sample from index start
* to index end.
*/
public static ArrayList<Integer> extractSample(ArrayList<Integer> sample, int start, int
end)
```