In this exercise you were given four methods to define, focusing on transforming values passed in to the methods through parameters to produce return values. There were two versions of this exercise – though the scenario was a little different in both cases the structure of the solution was the same in both.

For version 1 the scenario was described this way:

```
/*
* GENERAL INTRODUCTION
*
* In this coding exercise you will use these concepts and skills introduced
* in lecture to solve some real-world problems:
*
*      variables
*      expressions
*      methods
*
*
* PROBLEM STATEMENT
*
* An on-line retailer ships goods to customers and charges for shipping by
*  weight (in grams).
*
* You will write four methods, whose inputs and outputs are described below,
* that will compute the shipping costs for a package that whose weight is
* measured in lbs and oz.
*
*/
```

METHOD 1

The first method was described as follows:

```
/*
* METHOD 1: Converting pounds and ounces to ounces
* 50 competency points
*
* This method takes in a weight expressed in pounds (lbs) and ounces (oz)
* and returns the equivalent weight in ounces.
*
* The inputs to the method are:
*             lbs - the number of pounds
*             oz  - the number of ounces
*
* A pound is equal to sixteen ounces.  To convert a weight expressed in
* pounds and ounces to just ounces, multiply lbs by sixteen and add oz.
* Return the result.
*
*/
```

You were given a "stubbed out" implementation of the method. A method stub is minimal implementation of a method that returns a default value. For example, for numeric types, the default value is zero, whereas for booleans it is false.

```
public int question1(int lbs, int oz) {
	return 0;  // change the return value!
}
```

The inputs to this method represent the weight of a parcel, expressed in terms of pounds (lbs) and ounces (oz). The method is supposed to return that same weight expressed in just ounces, with the understanding that each pound corresponds to sixteen ounces.

For example, if a parcel weights 2 lbs and 3 oz, then the method should return 35 oz, because 2 lbs correspond to 2 x 16 = 32 oz, and adding the extra 3 oz produces 35.

Similarly, if a parcel weights 1 lb and 15 oz, then the method should return 31 oz, because 1 lb correspond to 1 x 16 = 16 oz, and adding the extra 15 oz produces 31.

In general, if a parcel weighs A lbs and B oz then the method should return A x 16 + B. The question is how to express this in code. In the method header we see that the two parameters are named `lbs` (corresponding to A in the example above) and `oz` (corresponding to B above). The computation will therefore be `lbs * 16 + oz`. Here's what the complete method will look like:

```
public int question1(int lbs, int oz) {
	return lbs * 16 + oz;
}
```

METHOD 2

The second method was described as follows:

```
/*
* METHOD 2: Converting ounces to grams
* 50 competency points
*
* This method takes in a weight expressed in ounces (oz)
* and returns the (approximate) equivalent weight in grams.
*
* The input to the method is:
*             oz  - the number of ounces
*
* An ounce is equal to 28.3495 grams.  Since we have not yet talked about
* numbers other than integers, we will approximate and say that one ounce
* converts as 28 grams.
*/
```

You were again given a "stubbed out" implementation of the method:

```
public int question2(int oz) {
       return 0;  // change the return value!
}
```

The input to this method is an int representing some number of ounces. The method is supposed to return the corresponding number of grams, under the assumption that each ounce corresponds to twenty-eight grams.

For example, 2 ounces correspond to 56 grams, because 2 x 28 = 56.

Similarly, 5 ounces correspond to 140 grams, because 5 x 28 = 140.

Generalizing, A ounces correspond to A x 28 grams. Again the question is how to express this in code. In the method header we see that the parameter is named oz (corresponding to A above). The computation will therefore be oz * 28. Here's what the complete method will look like:

```
public int question2(int oz) {
       return 28*oz;
}
```

METHOD 3

The third method was described as follows:

```
/*
* METHOD 3: Computing a shipping cost
* 50 competency points
*
* This method computes the shipping cost for a package of a given weight,
* expressed in grams.  The vendor imposes a minimum charge of $2.00 (which
* we will express as 200 cents.
*
* This method takes in a weight expressed in in grams (g) and returns the
* shipping cost in cents.
*
* The input to the method is:
*             g  - the number of grams
*
* The total shipping cost will be $2.00 plus one cent for each gram.  For
* example, a 3 gram package will cost 203 cents to ship, whereas a 250 gram
* package will cost 450 cents.
*
*/
```

You were again given a "stubbed out" implementation of the method:

```
public int question3(int g) {
       return 0;  // change the return value!
}
```

The input to this method is an int representing some number of grams. The method is supposed to return the shipping cost, which is described as consisting of a fixed charge of 200 cents plus one cent per gram of weight.

For example, a 56 gram package will be charged 200 + 56 x 1 = 256 cents.

Similarly, a 140 gram package will be charged 200 + 140 x 1 = 340 cents.

Generalizing, shipping a parcel weighing G grams will cost 200 + G x 1 cents. We can simplify this a little, to just be 200 + G, since multiplying by 1 does not change the result. In code, since the parameter is `g` the computation will be `200 + oz`. Here's what the complete method will look like:

```
public int question3(int g) {
       return 200 + g;
}
```

METHOD 4

The fourth method was described as follows:

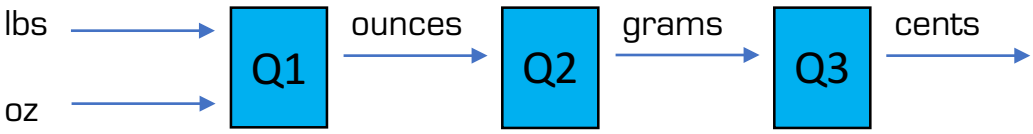
```
/*
* METHOD 4: Computing shipping cost from pounds and ounces
* 2 proficiency points
*
* This method computes the shipping cost for a package whose weight is
* expressed in pounds and ounces, and ALSO prints the shipping cost to the
* console, following the format of this example, for a weight of 3 lbs and
* 7 oz:
*
* The cost to ship a package that weighs 3 pounds and 7 ounces is 1740 cents.
*
*/
```

Here's the provided method stub:

```
public int question4(int lbs, int oz) {
	return 0;  // change the return value!
}
```

The inputs to this method represent the weight of a parcel, expressed in terms of pounds (lbs) and ounces (oz). The method is supposed to return the shipping cost, in cents, and also print out the cost in a given format. Let's tackle computing the shipping cost first.

Starting with a weight in lbs and oz we can use method of question 1 to give us a weight in oz, then use method of question 2 to produce a corresponding weight in grams, and finally use method from question 3 to compute a shipping cost in cents. Conceptually:



There are several ways we could express this in code. For example:

```
public int question4(int lbs, int oz) {
	int ounces = question1(lbs,oz);
	int grams = question2(ounces);
	int cents = question3(grams);
	return cents;
}
```

We can also string the method calls together, letting the result from one call feed directly into the next call:

```
public int question4(int lbs, int oz) {
	int cents = question3(question2(question1(lbs,oz)));
	return cents;
}
```

The method is also supposed to print out the shipping cost in a given format. The keys here are to compose the output using a combination of `System.out.print` and `System.out.println` statements, as well as positioning these statement beween the computation of the final value of `cents` and the return statement:

```
public int question4(int lbs, int oz) {
	int cents = question3(question2(question1(lbs,oz)));
	System.out.print("The cost to ship a package that weighs ");
	System.out.print(lbs);
	System.out.print(" pounds and ");
	System.out.print(oz);
	System.out.print(" ounces is ");
	System.out.print(cost);
	System.out.println(" cents.");
	return cents;
}
```