AP Computer Science A
Problem Set 0

Dr. Balchunas

Instructions

In this problem set, name your file FirstNameLastNamePSet0.java. For example, my file would be AndrewBalchunasPSet0.java. Therefore my driver class would also be named AndrewBalchunasPSet0. Questions will be answered inside of methods named by the example code after the question. Do **NOT** rename these methods or change their headers in any way. However, you may choose to create other helper methods that are called by these methods. You might want to test your methods by using a main method, but be sure to comment out your main method before submitting the assignment. Be careful that some questions ask for you to print code and others ask you to return a value. If your code does not run, it will not receive credit. It is ultimately up to you to decide the return and parameter types.

Problem 1

Given two ints a and b, return their sum. Unless the two values are the same, then return double their sum.
Example input/output:

```
sumDouble(a, b);
sumDouble(5, 5); //returns 20
sumDouble(1, 3); //returns 4
```

Problem 2

Given 2 ints a and b, return `true` if they are both in the range 20 to 30 inclusive, or they are both in the range 90 to 100 inclusive.
Example input/output:

```
inRange(a, b);
inRange(20, 31) //returns false
inRange(20, 21) //returns true
inRange(90, 100)//returns true
```

Problem 3

Given an integer x, return the absolute value of x.
Example input/output:

```
absolute(x);
absolute(-8); //returns 8
absolute(9);  //returns 9
absolute(-7); //returns 7
```

Problem 4

Given 2 ints a and b, return whichever value is nearest to the value 10, or return 0 in the event of a tie.
Example input/output:

```
closeNum(a, b);
closeNum(8, 13); //returns 8
closeNum(9, 11);  //returns 0
closeNum(7, 15); //returns 7
```

Problem 5

A harshad number is one that is divisible by the sum of its digits. Given any two digit number x, return `true` if it is a harshad number and `false` if it is not. *Hard mode: write the code so it works for integers of any length.*
Example input/output:

```
isHarshad(x);
isHarshad(24); //returns true
isHarshad(11); //returns false
```

Problem 6

Write a method that takes three integers as parameters and prints those numbers in ascending order.
Example input/output:

```
leastToGreatest(a, b, c);
leastToGreatest(4, 5, 1); //prints "Ascending order: 1, 4, 5"
```

Problem 7

Write a method that takes a double `d` as a parameter and returns the number rounded to the nearest integer.
Example input/output:

```
roundNum(x);
roundNum(1.6);  //returns 2
roundNum(-2.4); //returns -2
```

Problem 8

Many years in the future you have twins that are now 6 years old. As you know, these twins are up to no good if both are smiling or neither are smiling. Given two boolean parameters *aSmile* and *bSmile* to represent whether or not each twin is smiling, write a method that returns `true` if the twins are up to no good or `false` otherwise.
Example input/output:

```
twinTrouble(aSmile, bSmile);
twinTrouble(true, false); //returns false
twinTrouble(true, true);  //returns true
twinTrouble(false, false);//returns true
```

Problem 9

A quadratic equation is given by the mathematical form $f(x) = ax^2 + bx + c$ where a, b, and c are constants. The roots of the function are solutions to the equation $ax^2 + bx + c = 0$. There are always two roots, given by the expression $\frac{-b \pm \sqrt{b^2-4ac}}{2a}$. The options are to have two distinct real roots, one double root, or no real roots (where $b^2 - 4ac < 0$). Write a method that prints the roots of a quadratic curve when given the parameters a, b, and c. You can choose to represent the imaginary roots with $i = \sqrt{-1}$ or you can report that the equation has no real roots. *Use* `Math.sqrt(x)` *to return* $\sqrt{x}$.
Example input/output:

```
quadRoots(a, b, c);
//x^2 - 4
quadRoots(1, 0, -4); //prints "Root1: x = 2.0, Root2: x = -2.0"
//x^2 - 2x + 1
quadRoots(1, -2, 1); //prints "Double root: x = 1.0"
//x^2 - x + 1
quadRoots(1, -1, 1); //prints "No real roots!"
```

Problem 10

The gravitational force between two objects is determined by the following equation, $F = \frac{Gm_1m_2}{r^2}$ where m_1 and m_2 are the masses of the two objects in kilograms [kg], r is the distance between the objects in meters [m], and G is Newton's gravitation constant equal to 6.67 x 10^{-11} in SI units. Write a method that calculates the force on two massive objects m_1 and m_2 at a distance of r. You know from physics class that the force on a $1kg$ object at sea level is $F = mg = 1 * g = 9.8$ N. Using $m1 = 1$, $m2 = 5.92\text{x}10^{24}$ kg and $r = 6.371\text{x}10^{6}$ m, test that you get the expected result.
Example input/output:

```
calcForce(m1, m2, r);
calcForce(200000, 10000, 5); //returns 0.005336
```

Problem 11

When two celestial bodies orbit one another they actually orbit a common point. In the case where one of the bodies is much larger than the other like the sun and the earth, the point they orbit actually lies within the sun so it appears that the earth revolves around the sun and the sun stays stationary. When the masses of the objects are comparable the two masses orbit a central point called the barycenter. The barycenter is the center of gravity of the two masses and can be easily calculated with both masses m_1, m_2, and the shortest distance between the bodies a. Write a method that returns the barycenter r_1 of the binary system using the equation $r_1 = \frac{am_2}{m_1+m_2}$. Note here that m_1 is the more massive of the two bodies and r_1 is the distance from the more massive body to the barycenter. If r_1 is less than the radius of the more massive body R_1 then the barycenter lies within the first body. The inputs should be in the order of m_1, m_2, a. The output does not need to be an integer.
Example input/output:

```
//note we will give the ratio of the masses with respect to the mass of the earth.
barycenter(m1, m2, a);
barycenter(1, .0123, 384000);          //returns approx 4670
barycenter(.0021, .000254, 19600);     //returns approx 2115
barycenter(333000, 1, 150000000);      //returns approx 450
```

Problem 12

You will be using a method to model the following equation that determines the height of an object at a given time t. $y(t) = \frac{1}{2}gt^2 + v_{y0}t + y_0$. Here $g = -9.8$ $[\frac{meters}{sec^2}]$, t is time in seconds, v_{y0} is the initial velocity in the y direction in $\frac{meters}{sec}$, and y_0 is the initial y position in meters. We can assume the object starts on the ground so y_0 is 0. The method you will write will take a parameter of initial velocity (in the y-direction), and time and will return the y-position of the object.
Example input/output:

```
yPosition(vel, t);
yPosition(9.15, 1); //returns 4.25
yPosition(10, 2);   //returns approx 0.4
```

Problem 13

Write a method that takes three int inputs a, b, and c, representing the side lengths of a triangle, and return the area of the triangle using Heron's formula: $area = \sqrt{(s(s-a)(s-b)(s-c))}$, where $s = (a+b+c)/2$. Use the triangle inequality

to determine if the three sides can actually make a triangle. If the three sides cannot make a triangle, return -1. Recall the triangle inequality says that in order to be a triangle the sum of any two side lengths must be greater than or equal to the length of the third side for all combinations of sides.
Example input/output:

```
triArea(a, b, c);
triArea(2,3,4);     //return 2.905
triArea(5,6,7);     //return 14.6969
triArea(1,1,10);    //return -1.0
```

Problem 14

A problem of interest to computer scientists is that of modeling molecular dynamics. That is, how do molecules interact with one another as they move around? In chemistry or physics you may have heard of the Ideal Gas Law which says that particles are non-interacting and may pass through one another. In the real world this is not often the case. Particles are incapable of occupying the same space (due to electrons being unable to overlap), and they are attracted through one another through dispersion forces, sometimes called van der Waals forces. A model that works particularly well for calculating the atomic forces on noble gasses or less reactive gasses, such as methane, is the Lennard-Jones force. In it's simplest form it is $f = \frac{48\epsilon}{r^2}[(\frac{\sigma}{r})^{12} - \frac{1}{2}(\frac{\sigma}{r})^6]$, where ϵ is the depth of the potential energy well, or the minimum energy of the system, r is the distance between the particles, and σ is the distance at which particle-particle interaction is 0. This can be estimated to be the particle size. Given a well depth ϵ in energy units, r and σ in similar units, return the force on the two particles.
Example input/output:

```
// What does this force look like as a function of distance (r) for given particles?
// the arguments are in the order: well depth, particle size, distance
calcLJ(eps, sig, r);
calcLJ(3, 1, 5);    //returns -.000184
calcLJ(3, 1, 1);    //returns 72
```