Preparation for Midterm 1 SOLUTIONS

# Unit 2 Processing basics

Write an if-statement that prints “Yes!” if the last 3 letters of String *s* are equal to “*ing*” and s is less than 10 characters, but prints nothing otherwise.

if ( s.substring( s.length()-3, s.length() ).equals(“ing”) && s.length() < 10 ) {

println(“Yes!”);

}

Write a loop that prints the numbers 0.25, 0.50, 0.75, 1.00, 1.25, .1.50,…, 4.75, 5.00.

for( float x = 0.25; x <= 5.00; x += 0.25 ) {

println(x);

}

Write a loop that adds the numbers 0.25, 0.50, 0.75, 1.00, 1.25, .1.50,…, until the total exceeds 100. (In other words, this is an arithmetic series with *a* = 0 and *d* = 0.25)

|  |  |
| --- | --- |
| //Solution using a while-loop  float sum = 0, x = 0.25;  while( sum <= 100 ) {  sum += x;  x += 0.25  } | //Another solution using a for-loop  float sum = 0;  for( float x = 0.25; sum <= 100; x+= 0.25 ) {  sum += x;  } |

Declare a 1-D array named *dailyTemperatures* that can store 30 integers.

Then fill it with 30 random integers between -25 and 40.

Next, write a loop that finds the minimum temperature in the array.

int[] dailyTemperatures = new int[30];

for( int i=0; i<30; i++ ) {

dailyTemperatures[i] = int(random(66)) – 25; //recall that *nextInt*(n) returns a random

} //int from 0 up to n-1.

int min = dailyTemperatures[0];

for( int i=1; i<30; i++ ) {

if( dailyTemperatures[i] < min ) {

min = dailyTemperatures[i];

}

}

println(min)

The 2-D array

8 2 1 0

4 9 1 4

5 7 6 4

can be created using the shortcut int[][] a = {{8,4,5}, {2,9,7}, {1, 1, 6}, {0, 4, 4}}; Write a nested loop that prints array a in rows and columns. Write another nested loop that calculates and prints the averages of each column.  
 int[][] a = {{8,4,5}, {2,9,7}, {1, 1, 6}, {0, 4, 4}};

for (int i = 0; i < a.length; i++) { //in this example, a.length is 3, which is the number of rows

for (int j = 0; j < a[0].length; j++) { // a.length[0] is 4, which is the number of columns

print(a[i][j] + "\t");

}

println("");

}

for (int j = 0; j < a[0].length; j++) { //why is j the outer loop variable this time?

float sum=0.0; //why must this be float and not int?

for (int i = 0; i < a.length; i++) {

sum += a[i][j];

}

println( sum/a.length ); //why do we use a.length and not a[0].length?

}

Fill in the return-types of these functions

float getVolume( Sphere s ) { … }

int getNumberOfVertices( Polygon p ) {…}

boolean isInMandelbrotSet( ComplexNum z ) { … }

String[] getRandomArrayOfWords( int numWords ) {…}

void plotPoint( int x, int y ) { … }

float getAverageValueInArray( float[ ] a ) { … }

String getLast2Letters( String s ) { … }

void printArray( int[ ] a) { … }

//A function in the Fraction class that would be used in main() as follows:

//Fraction f = new Fraction(3,4);

//Fraction g = f.multByWholeNumber(5)

//g.display() 🡪 prints “15/4”

Fraction multByWholeNumber( int x ) { … }

Describe one major difference between Processing functions and functions in mathematics,   
like *f*(*x*) = 4*x*2

In math functions, the independent variables (called “parameters”or “inputs” in programming) and the dependent variables (called “return values” or “outputs”) are always numbers. Processing functions have no such restriction. Their parameters and return values can be numbers, of course, but also booleans, strings, and even complicated objects such as Scanners, or arrays of all of the above. Thus, Processing functions are much broader in scope than functions in mathematics.

Write a Processing function for the function *f*(*x*) = 3*x*4 + sin(*x*).

float **f**( float x ) {

return 3\*pow(x,4) + sin(x);

}

Write a function named *getShorter* that takes two Strings as parameters and returns the one that is shorter. In the main function, test the function on “Star Wars VII” and “The Martian”.

String **getShorter**( String a, String b ) {

if (a.length < b.length) {

return a; }

else {

return b;

}

}

Write a function named *getPolynomialValue* that takes two parameters:

* an integer array containing the polynomial’s coefficients in order from highest degree to lowest degree.
* an x-value

and returns the value of the polynomial at x. For example,

int[] p = {1, 2, 0, -5, 2}; //represents p(x) = 1x4 + 2x3 – 5x + 2, where the 0 means x2 is skipped.

float y = getPolynomialValue(p*,* 3); //returns 122, since **3**4 + 2(**3**)3 – 5(**3**) + 2 = 122.

float **getPolynomialValue**( int[] p, int x) {

float y = 0; //The y-value that will get returned

int degree = p.length - 1; //In the example above, degree = 5 – 1 = 4

for (int i = 0; i < p.length; i++) {

y += p[i] \* pow( x, degree-i ); //Can you see why the exponent is **degree-i**?

}

return y;

}

# Unit 4 Object oriented

Add a *divide* function to the *Fraction* class. Then use it in the *FractionTester* class   
to find and display the result. It should work very similarly to the *multiply* function.

|  |  |
| --- | --- |
| //Acceptable solution  Fraction **divide**( Fraction other ) {  int n1, n2, d1, d2;    n1 = this.numerator;  d1 = this.denominator;    n2 = other.numerator;  d2 = other.denominator;    return new Fraction( n1\*d2, d1\*n2);  } | //More elegant solution  Fraction **divide**( Fraction other ) {  Fraction r = other.getReciprocal();    return this.multiply( r );  }  //This is cooler because it makes use of other //functions we’ve already written. |

Create an animation of a ball flying across the screen up and to the right. If you’re bored, add a *gravity* variable which when set to something greater than 0 results in the ball flying in a parabolic trajectory.

void setup {

int f = 0; //frame counter

if(y <= 700 ) { //keeps the animation going for as long as y is still above "ground level"

x = 5\*f + 100; //updates x

y = gravity\*f\*f - 10\*f + 700; //updates y...try it with gravity = 0

f++; //increases the frame counter

}

}

Here is a design for a *Triangle* class. Study it, and then answer the questions that follow.

|  |  |
| --- | --- |
| **Fields** | **Description** |
| float ***x1, y1, x2, y2, x3, y3*** | The (x,y) coordinates of the three vertices |
| float ***A, B, C*** | The angles at vertices (xA, yA), (xB, yB) and (xC, yC) in degrees |
| float ***a, b, c*** | The side lengths opposite the angles ***A, B*** and ***C*** |
| String ***sideType*** | Whether the triangle is equilateral, isosceles or scalene |
| String ***angleType*** | Whether the triangle is acute, obtuse or right |

|  |  |
| --- | --- |
| **Functions** | **Description** |
| **Triangle**( float x1, float y1,   float x2, float y2,   float x3, float y3) | Constructor.  It should set the fields ***xA, yA, xB, yB, xC, yC*** using the parameters passed in from **main**().   Then it should set the fields ***a, b, c, A, B, C, sideType*** and ***angleType*** using the functions below. |
| float **getDistance**( float x1, float y1,   float x2, float y2) | Returns the distance between any two points (x1, y1) and  (x2, y2) using the length-of-a-line-segment formula |
| void **setLengths**() | Sets the values of ***a, b, c*** using the ***getDistance*** function and the values ***xA, yA, xB, yB, xC, yC*** |
| void **setAngles**() | Sets the values of ***A, B, C*** using the values ***a, b, c*** |
| void **setSideType**() | Sets the value of ***sideType*** using the values ***a, b, c*** |
| void **setAngleType**() | Sets the value of ***angleType*** using the values ***A, B, C*** |
| float **getPerimeter**() | Returns the perimeter of the triangle. It does not need any parameters because \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| float **getArea**() | Returns the area of the triangle. (Coding this requires some planning on paper since the base & height of the triangle have to be calculated using the values ***a, b, c, A, B, C****.*) |
| boolean **isSimilarTo** (Triangle other) | Returns true if *this* is similar to Triangle *other*, and false otherwise. |

One of the functions in the *Triangle* class should be *static*. Which one is it, and why? How would you call that function inside the *main* function of another class?

getDistance() should be static because it does not rely on any of the current triangle’s field values to do its job. It’s just used as a helper function by other functions in the class. (For instance, getPerimeter() would call getDistance() 3 times.)

If we wanted to use the getDistance() function from the main() function of some other class such as MyTriangleApplication, we would type something like

float d = Triangle.getDistance(5, 10, 0, 7); //Use the name of the class (Triangle) rather than

//the name of a particular Triangle object

Code as much of the *Triangle* class in Processing as you can. Then test the functions on a few sample triangles in a main class named *TriangleApp.*

See Unit 4/Lesson 4

Make a class named *BilliardBallsPainter* that draws this diagram. The *draw* function should use a nested loop. You’ll need to do some calculations on paper first.

I’m too lazy to code this right now. Maybe next semester’s students will have better luck. Sorry.