1. Given the following javadoc specification:
   static int min(int a, int b)
   Returns the smaller of two int values.

   a. [5 points] Show how to use the min() function to find the smaller of two integer variables
      x and y.

      int x, y, z;
      z = min(x, y);

   b. [5 points] Show how to use the min() function to find the smallest of three integer variables
      x, y, and z.

      int w, x, y, z;
      w = min(x, min(y, z));

2. [10 points] Write a method called issqrt() that takes two integer parameters (call them
   a and b) and returns true or false depending upon whether or not a is the square root of b. Do not use any other
   methods in your method.

   boolean issqrt(int a, int b) {
      return a*a == b;
   }

   or

   boolean issqrt(int a, int b) {
      if(a*a == b)
         return true;
      else
         return false;
   }
3. [10 points] Given the following numbered lines of code, write the numbers of the lines that constitute the scope of each variable.

```
1 class Bar {
2   public static void main(String[] args) {
3       int i = 5;
4       int j = 0;
5       while(j < 100) {
6           j = i*j;
7       }
8   }
9
10   { int k;
11     k = foo(i);
12   }
13   System.out.println(k);
14 }
15
16 public static double foo(int m) {
17     double n = 1/m;
18     return n;
19 }
20 }
21 }
```

i: Lines 4-15
j: Lines 6-8
k: Lines 11-13
m: Lines 17-20
n: Lines 18-20

4. [10 points] What does this program print out?

```
class TestProgram{
    public static void main(String[] args){
        int a = 1, b = 2;
        System.out.println(a);
        System.out.println(b);
        a = foo(a, b);
        System.out.println(a);
        System.out.println(b);
    }
}
```

static void foo(int x, int y) {
  x = 3;
  y = 4;
  System.out.println(x);
  System.out.println(y);
  return y;
}

It prints out:
1
2
3
4
4
2
5. Implement a method called `tn()` that takes an integer parameter `n` and returns $2^n$.

a) [10 points] Write a non-recursive implementation of `tn()`.

```c
int tn(int n) {
    int result = 1;
    if(n < 0)
        return -1;
    for(int i = 0; i < n; i++)
        result = result * 2;
    return result;
}
```

b) [10 points] Write a recursive implementation of the same function. Recall that $2^n = 2 * 2^{n-1}$.

```c
int tn(int n) {
    if(n <= 0)
        return 1;
    else
        return 2*tn(n-1);
}
```
6. Arrays
a) [5 points] Declare and create storage for an array of 11 ints called temps

    int[] temps = new int[11];

b) [10 points] Write a method called smooth() that takes an array of ints and finds the average of the numbers in the array, then adds 1 to each array element below the average and subtracts 1 from each array element above the average.

    void smooth(int[] a) {
        double avg = 0;
        for(int i = 0; i < a.length; i++)
            avg += a[i];
        avg /= a.length;
        for(int i = 0; i < a.length; i++)
            if(a[i] > avg)
                a[i]--;
            else if(a[i] < avg)
                a[i]++;  
    }

c) [5 points] Show how you would call smooth() with the temps array you created as a parameter

    smooth(temps);
7. [10 points] Write a method called `add()` that takes two integer arrays (of the same length) as parameters and returns a third array, where each element of the third array contains the sum of the corresponding elements of the two input arrays.

```java
int[] add(int[] a, int[] b) {
    int[] c = new int[a.length];
    for(int i = 0; i < c.length; i++)
        c[i] = a[i] + b[i];
    return c;
}
```

8. [10 points] Write a method called `filter()` that takes as a parameter a 2D array of integers, and returns an array in which each element in the original array has been replaced by \((u - d)/2\), where \(u\) is the element of the array above the current one (i.e. in the same column but the previous row) and \(d\) is the element of the array below the current one (i.e. in the same column but in the next row). Since this won’t work on the top and bottom rows, set them to all 0s.

```java
int[][] filter(int[][] a) {
    int[][] b = new int[a.length][a[0].length];
    for(int i = 0; i < a.length; i++) {
        for(int j = 0; j < a[i].length; j++) {
            if(i == 0 || i == a.length-1)
                b[i][j] = 0;
            else
                b[i][j] = (a[i-1][j] - a[i+1][j])/2;
        }
    }
    return b;
}
```
Extra credit. [10 points] Write a method called `am()` that multiplies two 2D matrices of numbers. It should take as parameters two 2D arrays of doubles, and return a 2D array of doubles representing their product.

The product C of two matrices A and B is defined as $c_{i,k} = \sum_{j=0}^{n-1} a_{i,j} \cdot b_{j,k}$.

It follows that if A is an $N \times M$ matrix and B is an $M \times P$ matrix, then C is an $N \times P$ matrix.

Graphically:

$$
\begin{bmatrix}
  c_{11} & c_{12} & \cdots & c_{1P} \\
  c_{21} & c_{22} & \cdots & c_{2P} \\
  \vdots & \vdots & \ddots & \vdots \\
  c_{a1} & c_{a2} & \cdots & c_{aP}
\end{bmatrix}
= 
\begin{bmatrix}
  a_{11} & a_{12} & \cdots & a_{1M} \\
  a_{21} & a_{22} & \cdots & a_{2M} \\
  \vdots & \vdots & \ddots & \vdots \\
  a_{a1} & a_{a2} & \cdots & a_{aM}
\end{bmatrix}
\times
\begin{bmatrix}
  b_{11} & b_{12} & \cdots & b_{1P} \\
  b_{21} & b_{22} & \cdots & b_{2P} \\
  \vdots & \vdots & \ddots & \vdots \\
  b_{a1} & b_{a2} & \cdots & b_{aP}
\end{bmatrix},
$$

where $c_{xy} = a_{x1} \cdot b_{1y} + a_{x2} \cdot b_{2y} + a_{x3} \cdot b_{3y} + \ldots + a_{xm} \cdot b_{my}$, for all $1 \leq x \leq N$ and $1 \leq y \leq P$.

```java
double[][] am(double[][] A, double[][] B) {
    double[][] C = new double[A.length][B[0].length];

    for(int x = 0; x < A[0].length; x++)
        for(int y = 0; y < B.length; y++)
            for(int k = 0; k < B[x].length; k++)
                C[x][y] += A[x][k] * B[k][y];
}
```