- **CODE EVALUATION.**

  For each of the following, give the value of the variable `x` after each line executes. If the line produces an error, then write `ERROR`. If the variable can have different values (as when using a random number generator), then indicate those values by writing, e.g., `x == 1 - 5`.

  i. `int x = 4 + 6 / 4 * 2 - 3;`

  ii. `int x = (int)(4.0 + 6.0 / 4.0 * 2.0 - 3.0);`

  iii. `int x = (int)( 3.5 / 6 );`

  iv. `int x = (int) 3.5 / 6 ;`

  v. `int x = (int) 3.5 / 6.0 ;`

  vi. `int x = 10 + (int) Math.random() * 10;`

  vii. `int x = 10 + (int)( Math.random() * 10 );`

  viii. `int x = 10 * (int)( Math.random() + 10 );`

  ix. `double x = 4 + 6 / 4 * 2 - 3;`

  x. `double x = 4 + 6 / 4.0 * 2 - 3;`

  xi. `double x = 4 + 6 / 4 * 2 - 3.0;`

- **CODE EVALUATION.**

  For each of the following, give the output of the code.

```
double[] d1 = { 1.0, 2.0, 3.0, 4.0, 5.0 };
double[] d2 = new double[d1.length * 2];
for ( int i = 0; i < d1.length; i++ )
{
    d2[i] = d1[i];
    d2[d2.length / 2 + i] = d2[i];
}

for ( int i = 0; i < d2.length; i++ )
    System.out.print( d2[i] + " " );
```

```
String[] words = { "Every", "good", "bird", "deserves", "feeding" };
for ( int w = 0; w < words.length; w++ )
{
    System.out.print( words[w] + ":" );
    for ( int j = 0; j < words.length; j++ )
        if ( words[w].length() < words[j].length() )
            System.out.print( " " + words[j]);
    System.out.println();
}
```

- **CODING NESTED LOOPS**

  Fill in the `main()` method in the class below so that when it runs it prints output (using `System.out.println()`) that looks like this:

```
5 4 3 2 1 0
4 3 2 1 0
3 2 1 0
2 1 0
1 0
0
```

  For full points, your code must use **nested loops**, each of which is actually used to generate the output. (You may use whatever types of loops you choose.)

```
public class Main
{
    public static void main( String[] args )
    {









    }
}
```

- **CODE COMPLETION.**

  Fill in the class below. Add the `sumArray()` method that has been called from the class constructor. When run, this method should act as follows:

  - If the arrays are of *identical* length, then it should return a new array of the *same* length, where each element is the sum of the elements at the same index in the input arrays. So, in the first call below, it would sum the first elements of the inputs, then their second elements, and so on, and output array `out1` would look like:

    $$\{2, 3, 5, 6, 8\}$$

  - If the arrays are of *different* lengths, then it will return an array, containing a number of entries equal to the length of the *shortest* array, summing only entries that appear in both.. Thus, for the second call, `out2` would look like:

    $$\{3, 4\}$$

```
public class Main
{
     public Main()
     {
        int[] arr1 = { 1, 2, 3, 4, 5 };
        int[] arr2 = { 1, 1, 2, 2, 3 };
        int[] arr3 = { 2, 3 };

        int[] out1 = sumArrays( arr1, arr2 );
        int[] out2 = sumArrays( arr2, arr3 );
     }









}
```