Indiana University
CSCI-C212: Introduction to Software Systems (S25) Lab 05
Joshua Crotts, Muazzam Siddiqui

Arrays & ArrayLists

What To Do:

Follow each step carefully. As you complete the lab, submit the source files (`.java`) problems to the autograder (link is in the Canvas portal). After finishing, please submit your work, as a ZIP, to Canvas, and let one of the AIs know.

For this lab, please place each method inside its own class file labeled as `ProblemX`, where `X` is the problem number. The accompanying test files should be named `ProblemXTest`.

Problem 1

Design the `int[] accSum(int[] A)` method, which receives an array of integers and returns a new array of integers that corresponds to the accumulated sum between each integer. We present some examples below.

```
accSum({1, 7, 2, 9})                => {1, 8, 10, 19}
accSum({1, 3, 3, 4, 5, 5, 6, 6, 2}) => {1, 4, 7, 11, 16, 21, 27, 33, 35}
accSum({5, 5, 5, 5, 5, 5, 5, 1, 5}) => {5, 10, 15, 20, 25, 30, 35, 36, 41}
```

Problem 2

Design the `int[] countEvenOdds(int[] vals)` method that returns a tuple (an array of two values) where index 0 stores the amount of even values and index 1 stores the amount of odd values.

```
countEvenOdds({11, 9, 2, 3, 7, 10, 12, 114}) => {4, 4}
countEvenOdds({11, 13, 15, 17})              => {0, 4}
```

Problem 3

Design the `boolean canBalanceArray(int[] A)` method, which determines whether or not an array of integers *A* can be split into a partition that balances each side. Use the following examples as motivation.

```
canBalanceArray(new int[]{1, 1, 1, 1, 5})    => false
canBalanceArray(new int[]{2, 3, 5})          => true
canBalanceArray(new int[]{-10, 2, -58, 50})  => true
canBalanceArray(new int[]{3, -1, -1, -1, 0}) => true
canBalanceArray(new int[]{3, 2, 1, 0})       => true
canBalanceArray(new int[]{10})               => false
```

Problem 4

A village has members where each has a partner. These members are grouped in pairs inside an `ArrayList<Integer>` where each pair of indices represents a relationship of the village. I.e., `A.get(`$2i$`)` and `A.get(`$2i$+1`)` are in a relationship. A couple is considered the wisest if they have the highest combined age. Design the `ArrayList<Integer> wisest(ArrayList<Integer> A)` method that, when given a list of (integer) ages A, return a new `ArrayList<Integer>` containing the ages of the wisest pair. If there is a tie, return the pair that has the highest age overall. The order is not significant. We present a few test cases below. You can assume that there will always be an even number of village members.

```
wisest({31, 42, 43, 35, 21, 27, 24, 44}) => {43, 35} or {35, 43}
wisest({47, 51, 52, 48, 33, 67, 45, 35}) => {33, 67} or {67, 33}
```