1_Passing Arrays to Functions: getData() and printData()

Whenever you need to read data and store it in a data structure, one of the initial tasks is to "get" the data (and store it in your data structure) and then "print" the data to make sure it is stored correctly. This lab asks you to write `getData()` and `printData()`, as well as determine simple statistics.

(0) Download the Lab4 Starter Kit from onCourse. SAVE the .zip on you Desktop and unzip the file.
(1) Create a new (console) project.
(2) Replace the default main.cpp with the files: **`arrays2functions.cpp`** and **`prototypes.h`.** *We'll come back to getStats.cpp later; you can ignore this file for now.*
(3) Read over the C++ code in **`arrays2functions.cpp`**. Study the code! Read the function prototypes carefully (in the file `prototypes.h`).
(4) **Fix the code** so it:
  (a) Calls `getData()` correctly to read data from the keyboard (standard input, `stdin`, keyboard) and fills the array (in the function `getData()`). You *must* fix the first line in the function to list the arguments correctly (Hint: see the function prototype).
  (b) Fix the comments in front of each argument, e.g., the first argument, the array, is an `/* out */` argument since the array arrives empty but returns filled with hmr values.

```
void getData( /* out  */ …
```

Note: arguments can be one of three types:
`/* in */` argument *arrives* with an existing value only (no new value is returned to the calling code).
`/* out */` argument *arrives* with no value (or with some bogus, default value) but is changed *in* this function and returns with a new value to the calling code.
`/* inout */` argument both arrives with an existing value and returns with an updated (new) value.

  (c) Fix the `printData()` function.
  (d) Document the arguments in `printData()` like you did in `getData()`.
  (e) Test the code by running it with various sets of inputs.
  (f) Call me over when this is working!

2_Passing Arrays to Functions: getStats()

(5) Add a new file `getStats.cpp` to your project. Notice that you now have two (2) `.cpp` files. One file contains `main()`, `getData()`, and `printData()` … while the second file contains only one function, the `getStats()` function. It is good software engineering to spread your functions over multiple .cpp files!
(6) Write complete documentation for the `getStats()` function.

```
// Summary: …
// PRE:
//
// POST:
//
// RETURN:
```

(7) Write documentation for the arguments (e.g., `/* in */`, etc.)
(8) **Fix** the `getStats()` function.
(9) Yup, test it! ***How do you really know that your answers are correct? How can you prove it?***
(10) **Using the Debugger:** Set a **breakpoint** at the start of `main()`. Practice **running your IDE *with the Debugger***. Make sure (a) you can see the values change in your local variables as you **STEP OVER**, (b) you can **STEP INTO** a function and check values there, and (c) you can view the runtime stack of function calls, for example, the function `main()` calls the function `getStats()`. **Call us over to verify that you can successfully use your IDE's Debugger.**