1 Arrays

1. What is the value of `result` when the following code is executed?

```
1        int[] taxicab = {1, 7, 2, 9};
2        int result = taxicab.length;
```

Solution: 4

2. What is the value of `result` when the following code is executed?

```
1        int[] taxicab = {1, 7, 2, 9};
2        int result = taxicab[1] + taxicab[3];
```

Solution: 7 + 9 = 16

3. What is the value of `result` when the following code is executed?

```
1        int[] taxicab = {1, 7, 2, 9};
2        int result = taxicab[taxicab.length - 1];
```

Solution: 9

4. What is the value of `result` when the following code is executed?

```
1        int[] taxicab = {1, 7, 2, 9};
2        int result = 0;
3        for(int i = 0; i < taxicab.length; i++) {
4                result+=taxicab[i];
5        }
```

Solution: 1 + 7 + 2 + 9 = 19

5. What is the state of array `taxicab` when the following code is executed?

```
1        int[] taxicab = {1, 7, 2, 9};
2        int result = 0;
3        for(int i = 0; i < taxicab.length; i++) {
4                taxicab[i]*=2;
5        }
```

Solution: $\{2, 14, 4, 18\}$

6. What is the state of array `taxicab` when the following code is executed?

```
int[] taxicab = {1, 7, 2, 9};
int result = 0;
for(int i = 0; i < taxicab.length; i++) {
        if(i % 2 == 0) {
                taxicab[i]*=2;
        }
}
```

$\{2, 7, 4, 9\}$

Solution:

7. What is the state of array `taxicab` when the following code is executed?

```
int[] taxicab = {1, 8, 6, 10, 9, 5, 7};
int result = 0;
for(int i = 0; i < taxicab.length; i++) {
        if(taxicab[i] % 2 == 0) {
                taxicab[i]/=2;
        }
}
```

Solution: $\{1, 4, 3, 5, 9, 5, 7\}$

8. Write a piece of code that declares and instantiates a array that can hold 8000 floating-point values.

Solution:

```
        double[] arr = new double[8000];
```

9. Write a piece of code that declares and instantiates an array `arr` that can hold 666 boolean values.

Solution:

```
boolean[] arr = new boolean[666];
```

10. Assuming that the array `arr` holds an array that holds 2000 integers (that is, it has already been declared and instantiated), write a piece of code, that, using a loop, assigns,

    - 1 to the first item of the array
    - 2 to the second item of the array
    - 3 to the third item of the array
    - ...

Solution:

```
int[] arr = new int[2000];
for(int i=0; i < arr.length; i++) {
        arr[i] = (i+1);
```

11. Assuming that the array `arr` holds an array that holds 2000 integers (that is, it has already been declared and instantiated), write a piece of code, that, using a loop, assigns,

    - 1 to the first item of the array
    - 5 to the second item of the array
    - 9 to the third item of the array
    - 13 to the fourth item of the array
    - ...

Solution:

```
int[] arr = new int[2000];
for(int i=0; i < arr.length; i++) {
        arr[i] = 1 + 4*i;
```

another way,

```
int[] arr = new int[2000];
int val = 1;
for(int i=0; i < arr.length; i++) {
        arr[i] = val;
        val+=4;
}
```

12. Assuming that the array `arr` holds an array that holds $n > 0$ integers (that is, it has already been declared and instantiated), write a piece of code, that, using a loop, assigns,

    - n to the first item of the array
    - $n - 1$ to the second item of the array
    - $n - 2$ to the third item of the array
    - ...
    - 1 to the last item of the array

    Note that you can access the number of items in array `arr` by `arr.length`.

    Solution:

```
int[] arr = new int[2000];
for(int i=0; i < arr.length; i++) {
        arr[i] = arr.length - i;
}
```

    another way,

```
int[] arr = new int[2000];
int val = arr.length;
for(int i=0; i < arr.length; i++) {
        arr[i] = val;
        val--;
}
```

13. Consider the following array `arr`,

```
        float[] arr=  {-1.2, 2.5, 1.3, 0, 0, 1.7, -1.9, 1.1, 0, 0, 0.6};
```

    (a) Write a piece of code that stores in a variable `result`, the number of items in array `arr` that are greater than 1.4.

    Solution:

```
int result = 0;
for(int i=0; i < arr.length; i++) {
        if(arr[i] > 1.4) {
                result++;
        }
}
```

    (b) Write a piece of code that stores in a variable `result`, the number of negative items in array `arr`.

    Solution:

```
int result = 0;
for(int i=0; i < arr.length; i++) {
        if(arr[i] < 0) {
                result++;
        }
}
```

(c) Write a piece of code that stores in a variable `max`, the highest value stored in array `arr`.

Solution:

```
int max = arr[0];
for(int i=1; i < arr.length; i++) {
        if(arr[i] > max) {
                max = arr[i];
        }
}
```

Note that the above method works only when the array has at least one item in it, otherwise generates `ArrayIndexOutOfBoundsException` if the array was instantiated to an array of size 0, or generates `NullPointerException` if the array has been initialised to `null`. The solution below works for **any** array.

```
int max = Integer.MIN_VALUE; //smallest value possible
if(arr != null) {
        for(int i=0; i < arr.length; i++) {
                if(arr[i] > max) {
                        max = arr[i];
                }
        }
}
```

14. Assuming that array `arr` hold 20 random integers, write a piece of code that stores in a variable `result`,

- `true` if the array `arr` is sorted in ascending order (such that each item is more than or equal to the previous item).
- `false` otherwise.

Solution:

```
boolean result = true; //assume to be in ascending order
for(int i=0; i < arr.length - 1; i++) {
```

```
3           if(arr[i] < arr[i - 1]) { //violation to ascending
                order
4                   result = false;
5           }
6   }
```

The above solution will go through the entire array even if the first two items are `arr[0] = 5, arr[1] = 2`. The following modification makes the loop immediately terminate as soon as the first violation is encountered.

```
1   boolean result = true; //assume to be in ascending order
2   for(int i=0; result == true && i < arr.length - 1; i++) {
3           if(arr[i] < arr[i - 1]) { //violation to ascending
                order
4                   result = false;
5           }
6   }
```

15. **(challenging)** Assuming that array `arr` hold 20 random integers, write a piece of code that stores in a variable `allUnique`,

    - `true` if every item in the array `arr` is unique.
    - `false` otherwise.

Solution:

```
1   boolean allUnique = true;
2   for(int i=0; allUnique == true && i < arr.length; i++) {
3           //check if arr[i] exists again
4      for(int k=i+1; allUnique == true && k < arr.length; k++) {
5        if(arr[i] == arr[k]) {
6          allUnique = false;
7        }
8      }
9   }
```