1 Functions - 2

1. Consider the following definition for function `foo`.

```java
int foo(int[] a) {
        int c = 0;
        for(int i = 0; i < a.length; i++) {
                if(a[i] == 0) {
                        c++;
                }
        }
        return c;
}
```

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

```java
        int[] arr = {6,0,9,0,0,8,0,8,9,7,0,0};
        int result = foo(arr);
```

Solution:

```
6
```

2. Consider the following definition for function `foo`.

```java
boolean foo(int[] a, int target) {
        for(int i = 0; i < a.length; i++) {
                if(a[i] == target) {
                        return true;
                }
        }
        return false;
}
```

(a) What is the value of `result` when the following code is executed?

```java
        int[] arr = {2,1,3,6,4,9,7,8};
        boolean result = foo(arr, 5);
```

Solution: `false`

(b) What is the value of `result` when the following code is executed?

```java
        int[] arr = {2,1,3,6,4,9,7,8};
        boolean result = foo(arr, 9);
```

Solution: `true`

3. Consider the following definition for function `foo`.

```java
boolean foo(int[] a) {
        for(int i = 0; i < a.length; i++) {
                if(a[i]%2 == 1) {
                        return false;
                }
        }
        return true;
}
```

Write a piece of code that calls the function `foo` by passing an integer array `arr` and storing the value returned in a variable `result`. You may assume the array `arr` has at least one item in it.

Solution:

```java
boolean result = foo(arr);
```

4. Consider the following definition for function `foo`.

```java
boolean foo(int[] a, int [] b) {
        if(a.length != b.length) {
                return false;
        }
        for(int i = 0; i < a.length; i++) {
                if(a[i] != b[i]) {
                        return false;
                }
        }
        return true;
}
```

(a) Write a piece of code that calls the function `foo` by passing two integer arrays `myArray, yourArray` and storing the value returned in a variable `result`. You may assume the arrays `myArray, yourArray` both have at least one item each.

Solution:

```java
boolean result = foo(myArray, yourArray);
```

(b) Explain briefly what is the purpose of function `foo`.

Solution: return `true` if both arrays are identical (in terms of content), `false` otherwise.

5. Write a function that when passed an array containing floating-point values, doubles each item of the array. Remember that modifying the formal array inside a function modifies the actual array passed.

Solution:

```
void doubleUp(float[] arr) {
        for(int i=0; i < arr.length; i++) {
                arr[i]*=2;
        }
}
```

6. Write a function that when passed an array containing floating-point values, negates each negative item of the array so that it becomes positive.

Solution:

```
void makeAbsolute(float[] arr) {
        for(int i=0; i < arr.length; i++) {
                if(arr[i] < 0) {
                        arr[i]*=-1;
                }
        }
}
```

7. Write a function that when passed an integer array, divides every even number in the array by 2.

Solution:

```
void divideEvensByTwo(float[] arr) {
        for(int i=0; i < arr.length; i++) {
                if(arr[i] % 2 == 0) {
                        arr[i]/=2;
                }
        }
}
```

8. **(somewhat challenging for most)** Write a function that when passed an integer array, resets any composite (non-prime) number to 0. You may define a *helper* method.

Solution:

```
boolean isPrime(int n) { //helper
        if(n < 2) {
                return false;
        }
        for(int i=2; i*i <= n; i++) {
                if(n%i == 0) {
                        return false;
                }
        }
        return true;
}

void resetComposites(int[] arr) {
        for(int i=0; i < arr.length; i++) {
                if(isPrime(arr[i]) == false) {
                        arr[i] = 0;
                }
        }
}
```

9. **(somewhat challenging for most)** Write a function that when passed an integer array, returns an array with only the positive items from that array.

Solution:

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

        int[] result = new int[countPositives];
        int targetIndex = 0;

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

        return result;
}
```