GROUP 2: Arrays

Note: You may not have used or even known about some of the interesting Java constructs below. Comments are provided so you can still figure out what is going on and there should be enough for you to figure out the time complexity of the running code in each case. It is therefore very important for you to read the comments.

a)

```
// retrieve n amount of values in O(n) time, i.e. getValues() takes linear
// time to execute. therefore n is the length of the values array below
int[] values = getValues();

// used to store the sum of all products
int allProducts = 0;

// loop through all values
// This uses an "enhanced for loop" used to go through all values in the
// array. Much like how Python's for loop works. The equivalent Python
// code would be "for value1 in values:" (without the quotes)
for(int value1 : values)
{
    // loop through all values for each value
    for(int value2 : values)
    {
        // calculate the product and store it in our running total
        allProducts += value1 * value2;
    }
}

// print everything out. Assume it takes constant time to print something
System.out.println("Sum of all products is " + allProducts);
```

b)

```
// retrieve n amount of values in O(n) time
// therefore n is the length of the values array below
int[] values = getValues();

// used to store the sum of some products
int someProducts = 0;

// loop through some values
for(int i = 2; i < values.length; i += 3)
{
    // loop through values for each value
    for(int j = values.length - 1; j > 1; j /= 5)
    {
        // calculate the product and store it
        // in our running total
        someProducts += values[i] * values[j];
    }
}

// print everything out
System.out.println("Sum of some products is " + someProducts);
```

c)

```
// retrieve n amount of values in O(n) time
// therefore n is the length of the values array below
int[] values = getValues();

// double and triple the value amounts
int[] doubleValues = new int[values.length];
int[] tripleValues = new int[values.length];

for(int i = 0; i < values.length; i++)
{
    doubleValues[i] = values[i] * 2;
    tripleValues[i] = values[i] * 3;
}

// used to store the sum of products
int totalProduct = 0;

// loop through all values in 'values'
for(int i = 0; i < values.length; i++)
{
    // loop through all values in 'doubleValues'
    for(int j = 0; j < doubleValues.length; j++)
    {
        // loop through all values in 'tripleValues'
        for(int k = 0; k < tripleValues.length; k++)
        {
            // calculate the product and store it in our running total
            totalProduct += values[i] * doubleValues[j] * tripleValues[k];
        }
    }
}

// print everything out
System.out.println("Total product is " + totalProduct);
```

d)

```
// retrieve n amount of strings in O(n) time
// therefore n is the length of the strings array below
// it is important to note that for this code, we will
// assume no single string is longer than n
String[] strings = getStrings();

// filter all strings by making a dynamic list and adding
// to it only strings that contain either an "a" or a "b"
// from our array
ArrayList<String> filteredStrings = new ArrayList<String>();
for(String string : strings)
{
    // even though the "contains" method runs a sequential search
    // over the string (as it looks for the target string within it)
    // note that the length of the string is NOT n, and we will
    // consider it to take constant time since it doesn't scale with n
    if (string.contains("a") || string.contains("b")) {
        filteredStrings.add(string);
    }
}

// print out all filtered strings
for(int i = 0; i < filteredStrings.size(); i++)
{
    System.out.println(filteredStrings.get(i));
}
```

e)

```
// for this code, assume we ALREADY HAVE an array called “amounts” that
// has been filled. So, without considering the time to create and fill
// such an array, analyze the time of just the code below:

if (amounts.length > 3)
{
    System.out.println(amounts[3]);
}
else if (amounts.length > 2)
{
    System.out.println(amounts[2]);
}
else if (amounts.length > 1)
{
    System.out.println(amounts[1]);
}
else if (amounts.length > 0)
{
    System.out.println(amounts[0]);
}
else
{
    System.out.println("empty array");
}
```