# CPS 112: Computational Thinking with Algorithms and Data Structures Homework 3 (20 points)

**Handed out:** February 14, 2018 (Wednesday)
**Due:** 11:59pm February 21, 2018 (Wednesday)
Late submissions accepted with penalty until 11:59pm February 23 (Friday)

**If you run into trouble or have questions, arrange to meet with me or the tutor!**

**Before you submit:**

- **Please review the homework guidelines available on Canvas**

- Read the assignment very carefully to ensure that you have followed all instructions and satisfied all requirements.

- This assignment involves no code! Write up your solutions in a PDF and submit it and your cover sheet directly to Canvas.

- Your writing should be of high quality. Your solutions should be polished, clear, precise, and thorough. Make sure you leave time to draft and revise. **The quality of your writing is a significant factor in your grade.**

- You may discuss the problems generally with your classmates but **do not collaborate on creating your write-up.** One good rule of thumb: do not write anything down during the discussion, and do not show your write-up to anyone. If you understand the discussion you should be able to clearly express the key ideas in your own words.

This assignment involves no programming.  Below you are given four functions and one or more example inputs.  For each function you should:

1. (2 pts) Using the provided example(s) show the step-by-step execution of the snippet of code by showing the changes in relevant values. After working through the example(s), give a concise description of what the code does. The key elements that will determine your grade:
   - Do you show all the steps?
   - Are the steps correct?
   - Is your description accurate?

2. (3 pts) Analyze the complexity of the function using big-O notation. You should clearly express your argument, analyzing increasingly large code blocks and explaining how you arrived at your conclusions (Similar to how it was done in lecture and problem 1 one the lab). The key elements that will determine your grade:
   - Is your overall complexity correct?
   - Is your reasoning clear? (Can I follow your logic?)
   - Is your reasoning thorough? (Have you considered all the important contributing factors?)
   - Is your reasoning compelling? (Do you justify your claims with well-presented evidence?)
   - Is your writing polished? (Is it well-written, and devoid of grammar and spelling errors?)

You should type your solutions to this homework and submit it as a PDF along with your coversheet. No other materials or files are necessary. To give you a sense of my expectations, see the example on the next page.

## Problem 0 (Example):

```
     int [] Mystery0(int [] a)
     {
1         int b = a[0];
2         int c = a[0];
3         int[] ans = new int[2];
4
5         for(int i = 1; i < a.length; i++)
6         {
7              if(a[i] < b)
8                   b = a[i];
9              else if(a[i] > c);
10                  c = a[i];
11        }
12        ans[0] = b;
13        ans[1] = c;
14
15        return ans;
     }
```

Example input: [2, 1, 5, 4]
Analyze complexity in terms of *n*, the size of *a*.

1. **Step-by-step execution**
   1. Lines 1-2: *b* = 2, *c* = 2
   2. Lines 5-11: *i* = 1, *a*[*i*] = 1 < *b*, so *b* = 1 (*c* stays the same)
   3. Lines 5-11: *i* = 2, *a*[*i*] = 5 > *c*, so *c* = 5 (*b* stays the same)
   4. Lines 5-11: *i* = 3, *a*[*i*] = 4. Since 4 < *c* and 4 > *b*, both *b* and *c* stay the same
   5. Line 5: *i* = 4, so the for loop stops
   6. Lines 12-13: *ans* = [1, 5]
   7. Line 15: *ans* = [1, 5] is returned

   This function finds the minimum and maximum value in an array.

2. **Analysis**
   Let *n* denote length of the array. The code block inside the loop (lines 7-10) only performs comparisons and assignments of integers. So this block is takes O(1) time. The loop itself iterates *n*-1 times (from *i* = 1 to *i* = *n*-1). Therefore the loop (lines 5-11) as a whole takes O(*n*) time. Lines 1-2 and 12-15 are simple assignments so they take O(1) time. Line 3 creates an array of size 2, regardless of *n*, so it takes O(1) time. Line 15 simple returns, also O(1) time. The loop is therefore the most expensive operation, and the function overall takes O(*n*) time.

## Problem 1:

```
void Mystery1(int[] a)
{
     for(int i= 0;  i < a.length - 1; i++)
     {
          for(int j = i + 1; j < a.length; j++)
          {
               if (a[i] > a[j])
               {
                    int tmp = a[i];
                    a[i] = a[j];
                    a[j] = tmp;
               }
          }
     }
}
```

Example input(s): [3, 6, 2, 1, 4]
[1, 2, 3, 7, 8]

Analyze complexity in terms of *n*, the size of *a*.

## Problem 2:

```
String Mystery2(String str, int n)
{
     String s = “”;
     for (int i = n; i > 0; i--)
     {
          s = s + str.substring(0, i);
     }
     return s
}
```

Example inputs: str = “Chocolate”, n = 4
str = “xyz”, n = 3

Analyze complexity in terms of *n* where *n* might be as large as the length of *str*. Assume `substring` is an O(n) operation

## Problem 3:

```
int Mystery3(int[] a, int x)
{
     int idx = -1;
     for(int i = 0; i < a.length && idx < 0; i++)
     {
          if(a[i] == x)
          {
               idx = i;
          }
     }
     return idx;
}
```

Example inputs: [4, 21, 9, 1], 5
[10, 2, 8, 24, 0], 8
Analyze complexity in terms of *n*, the size of *a*.

## Problem 4:

```
void Mystery4(int [] nums)
{
     int a = 0;
     for(int i = 0; i < nums.length; i++)
     {
          for(int j = nums.length; j >= i; j--)
          {
               if(nums[i] == nums[j])
               {
                    if (j – i + 1 > a)
                         a = j – i + 1
               }
          }
     }
     return a;
}
```

Example inputs: [3 , 3]
[1, 2, 3, 1]
[1, 4, 2, 1, 4, 3, 4]
Analyze complexity in terms of *n*, the size of *nums*.