**COMP 1917 Computing 1  
Session 2, 2014**

**Tutorial - Week 6**

**This page was last updated: 08/28/2014 12:22:17**

**Reminder - Prac Exam**

The first Prac Exam is this Friday (September 5) at 4pm. Please check the Prac Exam Instructions and [seating allocation](http://www.cse.unsw.edu.au/~cs1917/14s2/prac/seating.html) on the Course Web site.

**Tutorial Presentation**

Explain what is meant by a "Buffer-Overrun Attack".  
What kind of code is susceptible to such an attack?  
How can you re-write your code to protect it from this threat?

1. Given these declarations
2. int n;
3. int \*p, \*q;
4. double x;
5. double \*r;

What will happen when each of the following statements is executed (in order)? If any of them cause an error, explain why.

a. p = &n;

b. \*p = 5;

c. \*q = 17;

d. q = p;

e. \*q = 8;

f. r = &x;

g. \*r = 3.0;

h. \*p = \*r;

i. p = r;

j. \*p = n;

k. n = x;

1. Given that p is a pointer to int, write a function which asks the user to enter a number n, then allocates space for an array of n integers, and assigns it to p.
2. Write a function
3. int non\_decreasing( int a[], int N )

which takes an integer N together with an array a[] of N integers and checks to see whether the items in the array are sorted in non-decreasing order (i.e. a[i] ≥ a[i-1], for 0<i<N). Your function should returns 1 if the items are in non-decreasing order, 0 otherwise.

1. Write a function
2. int findIndex( int x, int a[], int N )

which takes two integers x and N together with an array a[] of N integers and searches for the specified value x within the array. Your function should return the smallest index k such that a[k] is equal to x (or -1 if x does not occur in the array).

1. Bonus Challenge: If the items in the array a[] are assumed to be in non-decreasing order, can you write an alternative version offindIndex() which runs much faster? (Hint: use the [Binary Search Algorithm](http://en.wikipedia.org/wiki/Binary_search_algorithm))

**Presentation Topic for Week 7**

Briefly describe the [Vigenere Cipher](http://en.wikipedia.org/wiki/Vigenere_cipher), and how it can be broken.