University of Windsor **COMP-1410 - Fall 2023** **School of Computer Science**

Midterm Practice Problems

1. Use recursion to write a function `count_ones` that returns how many 1s there are in a number n when represented in decimal (base 10). For example, 1231 has two 1s. You can assume that n is nonnegative and at most 9 digits long. Do not use global (or static) variables.

   In `main` perform at least three tests of `count_ones` and use `assert` to check that the returned value is correct. Your function should have the following prototype:

```
// count_ones(n) returns the number of 1s in the decimal
//   representation of n
// requires: 0 <= n < 10^9
int count_ones(int n);
```

2. Write an another implementation of `count_ones`, this time using looping (iteration).

3. Write a function `even_odd_sum` that takes as input an array and computes its "even-sum" and "odd-sum" where the even-sum is the sum of its even entries and its odd-sum is the sum of its odd entries. For example, for the array containing the entries $[1, 3, 2, 4, 5]$ the even-sum is $2 + 4 = 6$ and the odd-sum is $1 + 3 + 5 = 9$. The function must conform to the following specification:

```
// even_odd_sum(A, n, even, odd) updates *even to be the even-sum
//   of A and *odd to be the odd-sum of A
// requires: A is of length n
//           even and odd point to memory that can be modified
void even_odd_sum(int A[], int n, int * even, int * odd);
```

4. Write a function `column_maxs` that accepts an $n \times 5$ matrix as a 2D array and computes an array containing the largest number in each column. For example, the matrix $\begin{bmatrix} 1 & 0 & 5 & 9 & 6 \\ 4 & 3 & 2 & 8 & 7 \end{bmatrix}$ has largest column entries $[4, 3, 5, 9, 7]$. Complete an implementation using the contract given below.

```
// column_maxs(A, n, maxs) updates the array maxs to contain the
//   maximums of the columns of the n x 5 matrix A
// requires: 1 <= n
//           maxs points to an array of size 5
void column_maxs(int A[][5], int n, int maxs[]) {
```