# UMass Boston CS 240
# Homework 5
# Due 11/11/2019 24:00

Make a subdirectory hw5 in your home directory for this assignment.
Write your code in `cntSort.c`. Compile and run as follows:

```
icc -Wall main.c cntSort.c -o cntSort -lm
./cntSort -m 64 -n 1048576 -s 2019
```

The m, n, and s are optional command line arguments. s means a seed for generate random numbers.

## 1 Couting Sort

You may have learned some sorting algorithms – such as bubble sort and quicksort – in CS 110 and CS 210. This homework is about *counting sort*.

Let $n$ be the number of elements to be sorted. Counting sort assumes the elements are integers between 0 and $m$, and $m$ is much smaller than $n$.
For example, let $m$ be 3, and $n$ be 10. Then the elements can be 0, 1, or 2, and there are ten of them. For example,

```
unsigned data[10] = {0, 2, 1, 1, 0, 2, 2, 0, 1, 1};
unsigned count[3];
```

Counting sort uses an array `unsigned count[m]` and initializes all elements in `count` to zero.

Then we iterate through the array `data`.
For every `data[i]`, we increment count[data[i]] by 1. After all that, we write 0 for `count[0]` times, 1 for `count[1]` times, and 2 for `count[2]` times, etc..

Bubble sort takes $O(n^2)$ time in the worst case, and quicksort takes $O(n \lg n)$ time on average. Loosely speaking, quicksort is faster than bubble sort. You will learn the big-O notation in CS 310. Counting sort takes $O(m+n)$ time. We say it is a linear time algorithm, which is, loosely speaking, faster than quicksort.

The code in `main.c` is the driver. You implement counting sort in `cntSort.c`. The driver runs your counting sort as well as `qsort()` in the standard library. It verifies that your code works correctly, and reports the runtime of both sorting methods.