# STAT 598W Homework 2

Due by February 17, 2016

We will follow the same grading system as Homework 1. However, in this homework, the following four problems are required: 2.2, 3.1, 3.4, 4.1.

Section 1, 2 and 3 are of great importance. So I recommend that you attempt all of them. Section 4 is quite hard, but 4.1, 4.2 and 4.3 are not that hard, so you should try to finish them. 4.4 and 4.6 are also not very hard. 4.5, 4.7 and 4.8 are the hardest, but they are very interesting problems to solve.

## 1 String

1. Implement strcpy, strcmp, strlen, strcat and strchr, as in cstring/string.h.

   Please use the same input and output type as are used in the cstring/string.h.

## 2 Sorting

**In this problem, please sort the given array to THE DESCENDING ORDER, instead of the ascending order as shown in class.**

**Each of your implementation should be a function of the following form:**

```
void NameOfSortingAlgorithm (int *p, int n) {...}
```

**Your function should sort p[0], ..., p[n - 1] such that** $p[0] \geq p[1] \geq ... \geq p[n-1]$

1. Implement selection sort, insertion sort, bubble sort.
2. Implement quick sort.
3. Implement merge sort.
4. Implement heap sort.
5. Implement radix sort.
6. Implement counting sort.

# 3 Recursion and Dynamic Programming

1. Implement "nRooks". That is, we hope to put $n$ rooks, who do not threaten each other, in a $n \times n$ chess board. Your function should print all the possible solutions.

2. Implement "Queens and Rooks". That is, we hope to put $n/2$ queens and $n/2$ rooks, who do not threaten each other, in a $n \times n$ chess board, where $n$ is even. Your function should print all the possible solutions.

3. Given an array $a_0, ..., a_{n-1}$, calculate the following:

$$\max_{0 \le p \le q \le n-1} \sum_{i=p}^{q} a_i.$$

   Note that your array might not be non-negative.

4. Calculate max draw down of a price process. The max draw down of $a_0, ..., a_{n-1}$ is defined as:

$$\max_{0 \le p \le q \le n-1} (a_p - a_q).$$

   Actually, this problem is equivalent to the previous problem. But it might be better to consider this problem separately.

# 4 Large integer Computing

We know that all variable types in C++ have limitations, so standard C++ cannot deal with calculation that involves large integers. In this exercise, we will represent each integer by an array of char variables. For example, 143 is represented as char a[4] = "143", i.e. a[0] = '1', a[1] = '4', a[2] = '3', a[3] = '\0'. Then, we will compute addition, subtraction, multiplication and division of such types of large integer. **In this problem, please only consider NON-NEGATIVE integers.**

1. Complete the following function to compare two integers, a and b. If $a > b$, return 1, if $a = b$, return 0 and if $a < b$, return -1. Note that this is **NOT** strcmp, because we are comparing two integers.

```
int compare (const char *a, const char *b) {...}
```

2. Complete the following function to add two integers, a and b. The result is stored in dist. You may assume that enough space has been reserved for dist.

```
void add (const char *a, const char *b, char *dist) {...}
```

3. Complete the following function to subtract a by b. The result is stored in dist. You may assume that enough space has been reserved for dist. In this problem, you may assume that $a > b$.

```
void subtract (const char *a, const char *b, char *dist) {...}
```

4. Complete the following function to multiply two integers, a and b. The result is stored in dist. You may assume that enough space has been reserved for dist.

```
void multiply (const char *a, const char *b, char *dist) {...}
```

5. Complete the following function to divide a by b. The result is stored in dist. You may assume that enough space has been reserved for dist. Note that the result should give $\left[\frac{a}{b}\right]$, where $[x]$ is the largest integer that is no greater than $x$.

```
void divide (const char *a, const char *b, char *dist) {...}
```

6. You may notice that the function to calculate factorial that we implemented in homework 1 has very strong limitation even when $n$ is not very large. Write a function, which that calculate factorial based on part (4).

7. We know that the definition of the base of nature logarithm is the following:

$$e = \sum_{n=0}^{\infty} \frac{1}{n!} = 1 + \frac{1}{1!} + \frac{1}{2!} + ...$$

In this exercise, we will combine part (1), (2), (5) and (6) to calculate $e$. To do this, we will approximate $[e10^N]$, for certain large positive integer $N$. Indeed, we have

$$e10^N = \sum_{n=0}^{\infty} \frac{10^N}{n!} \approx \sum_{n=0}^{N'} \frac{10^N}{n!}.$$

The error term can be controlled as below:

$$\sum_{n=N'+1}^{\infty} \frac{1}{n!} \leq \frac{1}{N'!} \left( \sum_{k=1}^{\infty} \frac{1}{(N'+1)^k} \right) \leq \frac{1}{N'!} \approx \left(\frac{e}{N'}\right)^{N'} \frac{1}{\sqrt{2\pi N'}}.$$

Our goal is that $\left[[e10^N] - \sum_{n=0}^{N'} \frac{10^N}{n!}\right] \leq 1$. So we will choose $N'$ such that $N'! > 10^N$. In this way, $\left[\sum_{n=0}^{N'} \frac{10^N}{n!}\right]$ yields the correct first $N$ digits of $e$ except the last digit.

8. We can also try to compute $\pi$. We first notice the following Taylor's expansion of $\arctan x$:

$$\arctan x = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + ...$$

Then, we have

$$\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + ...$$

However, such series converges quite slow. The convergence rate is $O(n^{-1})$. A way to improve is by using the following formula for $\pi$:

$$\frac{\pi}{4} = 4 \arctan \frac{1}{5} - \arctan \frac{1}{239}.$$

The formula can be proved by involving complex number. Define $z_1 = 5 + i, z_2 = 239 + i$. Then, we have

$$z = \frac{z_1^4}{z_2} = \frac{476 + 480i}{239 + i} = \frac{114244 + 114244i}{57122}.$$

Therefore, we have

$$\frac{\pi}{4} = \arg z = \arg \frac{z_1^4}{z_2} = 4 \arctan \frac{1}{5} - \arctan \frac{1}{239}.$$

Note that in this way, the rate of convergence is changed to $O(5^{-n})$.