STAT 5443: Homework 1

Due: 17th February 2017

1. This homework has 4 problems and a total of 40 points, and it will contribute 10 points towards your final score (the lowest homework score will be dropped).
2. Attempt as many problems as possible.
3. Only neatly handwritten solutions will be accepted. Alternatively you may use LaTeX to typeset your solutions.
4. Hand in your HW (including print outs of your source code) at the beginning of the class on 17th February 2017. Additionally source code (if any) should be emailed to `stat5443.fall@gmail.com`. **before** the assignments are submitted in the class. No late submissions will be accepted!
5. Program files should be named after the problem (e.g. solution to problem 1 should be problem1.R etc).

Problem 1 ($4 \times 2 = 8$ **pt)** Write a R program which does the following in sequence:

- Creates a vector `x` which contains the numbers $[9, 10, 11, 12, 13, 14, 15, 16]$.
- Prints the last three elements of `x`.
- Prints out all the even numbers in `x`.
- Deletes all the even numbers from `x` and prints the resulting list.

Problem 2 ($3 \times 4 = 12$ **pt)** Write a R program which takes as input an integer `n` and computes $\sum_{i=1}^{n}(1/2)^i$ in three ways: using a `for` loop, using a `while` loop, and without using a loop (e.g., by using an analytic formula). Print out all three results. What happens when `n` is very large? Do you have any suggestions on how to make your program more robust to errors when `n` is large?

Problem 3 ($5 \times 2 = 10$ **pt)** Show that

- x^3 is $O(x^3)$ and $\Theta(x^3)$ but not $\Theta(x^4)$
- For any real constants a and $b > 0$, we have

$$(n + a)^b = \Theta(n^b).$$

- $(\log(n))^k = O(n)$ for any k
- $n/(n + 1) = 1 + O(1/n)$
- $\sum_{i=0}^{\lceil \log_2(n) \rceil} 2^i$ is $\Theta(n)$

Problem 4 (10+5+5=20 pt) We want to compare the relative time consumption by **merge sort** and **bubble sort** in this exercise. Use the R code for merge sort posted on blackboard or use your own implementation of mergesort. Use R to conduct the following:

(a) Implement the selection sort algorithm taught in the course.

(b) Generate a random sample of size 100 from the normal distribution with mean 0 and variance 1. The R command for generating a random sample from the normal distribution is `rnorm(n, mean = 0, sd = 1)`. Run both `mergesort` and `bubblesort` for this input and compare the CPU times (e.g. using `proc.time`).

(c) Repeat the part (a) 500 times, resulting in 500 cpu-times. Calculate the sample mean, and sample variance of the 500 cpu-times for both the sorting algorithms and compare them. Do the results agree with your expectation? Explain.

Hint: You can use loop in R for part (b) to save the 500 proportions. Here is an example code:

```
time1 = rep(0,500); time2 = rep(0,500)
for(i in 1:500){
  #obtain the cpu time from the i-th random sample
  #and save the results as, say, t1 and t2
  y =  rnorm(100) #generated sample of size 100
  ptm <- proc.time()
  mergesort(y) #merge sort algorithm
  t1 = proc.time()-ptm
  time1[i]=t1[["elapsed"]]
  ptm <- proc.time()
  bubblesort(y) #selection sort algorithm
  t2 = proc.time()-ptm
  time2[i]=t2[["elapsed"]]
}
```