**Programming Assignment #3**

**1.**Download the following text file:

**Median**

TXT File

[Download file](https://d3c33hcgiwev3.cloudfront.net/_6ec67df2804ff4b58ab21c12edcb21f8_Median.txt?Expires=1634860800&Signature=ULuTeBKciDob5NuTAylnjkryA16RooM7-Viwi11s80DeQ1wLEME1mutXAoHJn--b~gnhF5x7T1G4dpkF-Xy-nt9ZgnPlsxAw5ln40oE8NNolRXnOKY6LuiMvjjd2XFib10d07dSMWQsiiHsXbMHNv0r12kWWSnRyQMLywhf9gfI_&Key-Pair-Id=APKAJLTNE6QMUY6HBC5A" \t "_blank)

The goal of this problem is to implement the "Median Maintenance" algorithm (covered in the Week 3 lecture on heap applications). The text file contains a list of the integers from 1 to 10000 in unsorted order; you should treat this as a stream of numbers, arriving one by one. Letting *xi*​ denote the *i*th number of the file, the *k*th median *mk*​ is defined as the median of the numbers *x*1​,…, *xk*​. (So, if *k* is odd, then *mk*​ is ((*k*+1)/2)th smallest number among *x*1​,…, *xk*​; if *k* is even, then *mk*​ is the (*k*/2)th smallest number among *x*1​,…,*xk*​.)

In the box below you should type the sum of these 10000 medians, modulo 10000 (i.e., only the last 4 digits). That is, you should compute (*m*1​+*m*2​+*m*3​+⋯+*m*10000​) mod 10000.

OPTIONAL EXERCISE: Compare the performance achieved by heap-based and search-tree-based implementations of the algorithm.