# Lab #6 CS 1021C-002 - Spring 2014

## Date:

Wednesday, February 19th, 2014

## Objective:

File Input/Output, Functions and Arrays.

## Lab Problem:

You will be reading in a file of integers (called input.txt). The file may be quite large, but that is not a problem since we won’t be storing all the numbers in memory.

Please perform the following tasks in order. You should verify that each step is working correctly before moving on to the next step.

### Input file

You should download the input.txt file from here:

<https://drive.google.com/file/d/0B-ZYwKXag4_xZFo5UEM2OFMxenM/edit?usp=sharing>

There is also a shorter version (short.txt) that you should start with and use for initial testing.

<https://drive.google.com/file/d/0B-ZYwKXag4_xMUNCVW9tSHNRWmM/edit?usp=sharing>

### Task 1

Write a program that opens the file, called input.txt, and reads in all of the integers in the file. You should stop reading in number when you find the sentinel value of -1. All you need to do for the first part is count the number of numbers that you have read in. Everything for this part should be in main.

Here is sample output from task 1. For all of the sample output, this is the correct output for the file provided.

$ ./a.out

Numbers read in: 100007

### Task 2

Write a function, called isNotEven() that takes in an integer using pass-by-value and returns a boolean value. The return value should be true if the number passed in is not even, and false if the number is not odd.

Inside the loop that is processing the file, you should call isNotEven for each number, and output the number of odd numbers.

Here is sample output from task 2.

$ ./a.out

Numbers read in: 100007

Odd numbers: 50140

### Task 3

Add a new function called isNotDiv2or3 that takes in an integer using pass-by-value and returns a boolean value. The return value should be true if the number passed in is not divisible by 2 or 3, and false if the number is even (divisible by 2) or if it is divisible by 3.

Sample output after task 4:

$ ./a.out

Numbers read in: 100007

Odd numbers: 50140

Not Divisible by 2 or 3 numbers: 33363

### Task 4

Add a new function, called isPrime(), that takes in a number and returns true if the number is not divisible by any smaller number, that is to say it is a prime number. Simply checking that no number between 2 and the squareroot of the number is a divisor can be used to determine whether a number is prime. Do you see why we use squareroot as upper bound?

Here is the sample output for the full input.txt file.

$./a.out

Numbers read in: 100007

Odd numbers: 50140

Not Divisible by 2 or 3 numbers: 33363

Prime numbers: 6779

### Task 5

Finding the top-k largest values in the input stream.

To do this, declare a constant int k (initially set to 10). Declare an array of integers of size k in the main function; you can call it the topk array. Initialize the array to all zeros.

Write a function that takes in an array, the size of that array, and the value currently being examined. Call this function in the same loop that you are calling all of the other functions in.

The function header will look like this:

void updateTopk(int topk[], int k, int x)

To determine the largest k numbers, we are going to use an insertion-style algorithm on the topk array. The algorithm works like this

* start with the array and the value x you want to insert
* compare the value x against the first position in array
  + if x is less then simply return with no updates
  + otherwise shift each array element less than x down one position
  + finally place x in its proper position, which we just freed up for x

Here is pseudocode for the function:

if (x < A[0]) return without any updates

else {

i = 0

while (x > A[i] and i < k) {

A[i]= A[i+1]

increment i }

A[i-1] = x }

So, to start out with, if the array has been initialized to all zeros, and you are given the value 5. You will test 5 against the first element (index 0), see that it is greater, and then begin shifting each of the 0 values down one position in array. Finally the 5 will be stored in the last position of the array.

Here is the expected output based on the input file given.

$ ./a.out

Numbers read in: 100007

Odd numbers: 50140

Not Divisible by 2 or 3 numbers: 33363

Prime numbers: 6779

Largest k = 10 numbers:

9998778 9998779 9998984 9999048 9999368 9999465 9999740 9999749 9999923 9999930

### Task 6

The goal is finding the top-k largest prime numbers in the input stream. Declare another array topkprimes of size k, and initialize values to 0. For every prime number in the input stream, call the function updateTopk(int [], int , int ) passing the topkprimes array and the prime number to potentially update the array.

Here is the expected output based on the input file given.

$ ./a.out

Numbers read in: 100007

Odd numbers: 50140

Not Divisible by 2 or 3 numbers: 33363

Prime numbers: 6779

Largest k = 10 numbers:

9998778 9998779 9998984 9999048 9999368 9999465 9999740 9999749 9999923 9999930

Top k = 10 prime numbers:

9988361 9990121 9993701 9993703 9994289 9994573 9995239 9995659 9998447 9999749