Due date:

This file contains Lab 5. You must submit your answers to the D2L Dropbox "Lab-5"

1. Answers to question 1 below, written on this file**.**

2. A working java program for question 2.

Lab 5 requires Java programming. You can work in pairs (but you must still submit your own work to D2L).

**Note that late assignments will not be graded.**

Please do not zip or compress your submissions. D2L allows you to upload multiple files

1- The Trans-Canada Highway (TCH) follows a completely straight line as it crosses the Canadian prairies. Gas stations occur at various intervals. Assuming that the first station is a mile 0, and the last station is at mile n, find the distance between two closest stations. (The distance between two stations x and y is computed as |x − y|.)

Input is the mile maker locations of the gas stations, for example:

480 231 0 477 121 ... 1176 501 2000 (for n=2000)

Output for this example (based on the input we can see) would be: |480 – 477| = 3

A). Design and write pseudocode for a presorting-based transform and-conquer algorithm that solve this problem. [2 mark]

//input sorted array first  
//input A[0..n-1] of orderable elements  
  
//output the smallest difference between two adjacent values  
  
smallestDifference(int array A[0..n-1])  
 int index\_first <- 1  
 int index\_second <- 0  
 int min <- (A[index\_first] - A[index\_second])  
 for (i <- 2; i < A.length; i <- i+1)  
 if (A[i] - A[i-1] < min) Then  
 index\_first <- i  
 index\_second <- i-1  
 min <- A[index\_first] - A[index\_second]  
 print("|" & A[index\_first] & " - " & A[index\_second] & " = " & min & "|")

B). Design and write pseudocode for a brute-force algorithm that solve this problem. [2 mark]

//input A[0..n-1] of elements  
  
//output the smallest difference between two consecutive values  
  
smallestDifference(int array A[0..n-1])  
 int index\_first <- 1  
 int index\_second <- 0  
 int min <- A[index\_first] - A[index\_second]  
 for (i <- 0; i < A.length; i <- i+1)  
 for (j <- i+1; j < A.length; j <- j+1)  
 //prevent negative values and repetition by adding > 0 constraint  
 if (A[j] - A[i] < min AND A[j] - A[i] > 0) Then  
 index\_first <- j  
 index\_second <- i  
 min <- A[index\_first] - A[index\_second]  
 print("|" & A[index\_first] & " - " & A[index\_second] & " = " & min & "|")

C). Compare the efficiency of your algorithm for part A with your algorithm for part B. [1 mark]

Brute force algorithm goes through i to n and i+1 to n, therefore its efficiency is n \* n

so O(

Presorting based algorithm uses merge sort to initially sort the arrays, then takes the sorted array and compares linearly the adjacent values.

Merge sort efficiency is O(n log n)

Linear compare efficiency is O(n)

(n log n) + (n) = (n log n)

So O(n log n)

2- Design and implement an algorithm that finds the smallest k numbers (in value) out of n numbers. For example, if given an array with eight numbers {4, 5, 1, 6, 2, 7, 3, 8}, return the least four numbers 1, 2, 3, and 4.

The algorithm that sort the n input numbers increasingly and returns the first K number is not acceptable for the answer. Since it needs to sort, its time complexity is. You should design more efficient algorithm using Max-heap. [5 mark]

Hint.

You should create a max-heap with capacity k that will contain the least k numbers out of n input numbers.

In Java, the PriorityQueue class is implemented as a priority heap. You can find a sample code in D2l that shows the basic operations of PriorityQueue such as offer(), peek(), poll(), and size().