CSCI 406: Algorithms
Solution to Assignment 4

4-2 **[10 pts]**

1. (a) let x be the max of the integers and y be the min of the integers. Both can be computed by a pass through the entire array in $\Theta(n)$ time. Return $|x - y|$.
2. (b) Let y be the leftmost element in the array and x the rightmost, both computable in $O(1)$ time. Return $|x - y|$.
3. (c) First sort the array in $O(n \log n)$ time, then continue with the solution described in part (d) below.
4. (d) Scan each pair of neighboring elements to find the pair with the smallest non-zero difference.

4-6 **[10 pts]**

1. First, sort array S_2 in $\Theta(n \log n)$ time.
2. For each element $y = S_1[i]$, for $i \in [1, n]$, perform a $O(\log n)$ time binary search in S_2 for the value $x - y$ for a total of $O(n \log n)$.

Hint: An alternative solution sorts both S_1 and S_2 and then performs a simultaneous $\Theta(n)$ time left-right scan of S_1 along with a right-left scan of S_2 to find a pair of numbers that adds to x

4-12 **[10 pts]**

Step 1: Build a min-heap in $O(n)$ time. (Use the BUILD_HEAP and HEAPIFY algorithms discussed in class to achieve the $O(n)$ time).

Step 2: Extract the minimum element from the heap k times in $O(k \log n)$ time (each EXTRACT-MIN takes $O(\log n)$ time).

The total time for the algorithm is $O(n + k \log n)$.

4-29 **[10 pts]**

We know the lower bound for sorting is $\Omega(n \log n)$.
Assume B.C. Dull is correct about his new data structure and that the data structure is based on direct *comparisons* between elements. Using his data structure we could sort in $O(n)$ time: First, insert all n elements into the priority queue in $O(n)$ time. Next, use Extract-Max to extract each element and place the extracted elements from right to left in an array in $O(n)$. The resulting array is sorted. This contradicts the lower bound for sorting, so Mr. Dull must be mistaken..

4-33 **[10 pts]**
Look at the median element i.
If $a_i = i$, then return true
If $a_i > i$, all elements i to n can be discarded

If $a_i < i$, all elements 0 to i can be discarded
Recursively repeat for the remaining elements
If there are no remining elements, return false.