Assignment 1: Sample solutions and comments

1. Hopefully everyone understood this question, so no further explanation is needed.

2. (a) Despite the hint, many people seemed somewhat confused about how to structure the induction argument. The basis of the induction is the case where $f(A, B, C) = 0$. Since both $|B|$ and $|C$ remain greater than zero after the first comparison, we can reach a state where $f(A, B, C) = 0$, only when $|A| = n = 1$, or $|A| = 0$ and $|B| = |C| = 1$. In either case, there remains just one candidate for each of max and min, so no more comparisons are needed (which establishes the basis of the induction).

   For the induction step, which most people handled correctly, we observe that for all possibilities of the next comparison (I will not go through all of them here), following the rules of the adversary ensures that $f(A', B', C')$, the value of f on the updated sets, is reduced by at most 1. (For example, if the next comparison is of type A:A, then $|A'| = |A| - 2$, $|B'| = |B| + 1$ and $|C'| = |C| + 1$, yielding $f(A', B', C') = f(A, B, C) - 1$.)

   If we describe a comparison as *productive* if it leads to a reduction in the value of f, then our induction hypothesis can be strengthened to assert that "In any state (A, B, C, D) at least $f(A, B, C)$ more productive comparisons are required to reach a state in which both the maximum and minimum element are known". If we proceed by induction on $f(A, B, C)$, we note that, after one productive comparison, we reach a state (A', B', C', D'), where $f(A', B', C') = f(A, B, C) - 1$, and so, by our inductive hypothesis at $f(A, B, C) - 1$ more productive comparisons remain to be done.

   (b) The lower bound follows by observing that at the start of the algorithm $|A| = n$ and $|B| = |C| = 0$.

3. (a) For this problem, you want to choose p_c to be the midpoint of the interval $[\min_S, \max_S]$, where $\min_S$ (resp., $\max_S$) corresponds to the minimum (resp., maximum) element in S. You need to argue that any other choice will be sub-optimal (could be improved); note that any other choice for p_c will be further from either $\min_S$ or $\max_S$.

   It is not completely clear that both $\min_S$ and $\max_S$ need to be determined by any algorithm that finds their midpoint (which would

establish a lower bound of $\lceil 3|S|/2 \rceil - 2$ comparisons in the worst case, by question 2). Nevertheless, it is clear that $\Omega(n)$ comparisons are needed, since we cannot determine the centre of a set without at least looking at all of its elements.

(b) For this problem, you want to choose p_c in such a way that the number of elements of S that are less than or equal to p_c matches the number of elements of S that are greater than or equal to p_c. (Otherwise, the expression $\sum_{q \in S} |q - p_c|$ can be reduced by either increasing or decreasing p_c.)

Thus, this problem reduces to finding the median, when $|S|$ is odd, (or a median, when $|S$ is even) element in S. You should recall (from CPSC 320) that this can be done in $O(n)$ time; that is, it is not necessary to sort the elements. The fact that $\Omega(n)$ comparisons are needed follows from the observation that we cannot determine the median of a set without at least looking at all of its elements.

(c) For this problem, you want to choose p_c to be either 0 or 1 or the midpoint of the largest gap between successive elements of S. (Again, you should be able to argue that no other point could realize the maximum value of $\min_{q \in S} |q - p_c|$.)

To choose the best among these possibilities it suffices to know the min and max values in S and the sizes of all the gaps between successive elements. Clearly, this can be determined by sorting the elements of S, but it is interesting to know that this is not necessary: the largest gap (like the min and max) can be found in linear time. We will talk more about how to do this later. Meanwhile, as in part (b) is should be clear that $O(n)$ time is the best one could hope for.

(d) For this problem, you want to choose p_c to be the midpoint of the *smallest* gap between successive elements of S. (Again, you should be able to argue that no other point could be the centre of a minimum length interval that covers two or more points of S.)

To choose the best value for p_c, it suffices to know the sizes of all the gaps between successive elements. Clearly, this can all be determined by sorting the elements of S, but as in part (c), it is interesting to know whether or not sorting is necessary. It turns out that the answer depends on the model of computation (allowable operations) in a subtle way. Again, we will talk more about this later.