Recall that your solution must be typed via Latex. Also, remember that this is a group assignment; group size is at most 2.

**Deliverables:** Submit to Canvas two PDF files: a writeup for the homework solution, and a report that describes how you discovered your solutions. Both files should contain the names of all group members.

1. (70 points) Let $H : \{1, \ldots, N + 1\} \to \{1, \ldots, N\}$ be a hash function that is *non-decreasing*, meaning that if $a \leq b$ then $H(a) \leq H(b)$. Since the domain size is bigger than the range size, by Pigeonhole’s Principle, there must be a collision. Design an algorithm to find a collision on $H$. Your algorithm should treat $H$ as a black box that it can evaluate on $O(\log(N))$ inputs. Do not make extra assumptions on the function $H$.

   [We expect an English description of your algorithm, and a short, informal argument for its correctness and running time.]

2. (70 points) Consider an $N$-node complete binary tree $T$, with $N = 2^n - 1$. Each node $v$ of $T$ is labeled with a real number $x_v$. You may assume that the real numbers labeling the nodes are all distinct. A node $v$ of $T$ is a *local minimum* if the label $x_v$ is less than the label $x_w$ for all nodes $w$ that are joined to $v$ by an edge.

   You are given such a complete binary tree $T$, but the labeling is only specified in the following *implicit* way: For each node $v$, you can determine the value $x_v$ by *probing* the node $v$. Show how to find a local minimum of $T$ using only $O(\log(N))$ probes to the nodes of $T$.

   [We expect an English description of your algorithm, and a short, informal argument for its correctness and running time.]

3. (70 points) You are given a number $x$ and an array $A[1 : n]$ of (possibly negative) numbers. Give a divide-and-conquer algorithm to check whether or not there’s a subarray $A[i : j]$ of $A$ such that $A[i] + \cdots + A[j] = x$. Your algorithm should run in $O(n \log(n))$ time.

   [We expect an English description of your algorithm, and a short, informal argument for its correctness and running time.]

   **Hint:** In solving a hard problem, you should always ask yourself, “Have I seen a similar problem before?” Here you can easily see that a related problem is the Maximum Subarray problem. But there’s *another* related problem in the scribes that will be useful here. You need both of these two problems in order to solve this question.