# Assignment 2

Due: January 23, 2020

Your solutions must be typed (preferably typeset in LaTeX) and submitted as a hard-copy at the beginning of class on the day its due.
When asked to provide an algorithm you need to give well formatted pseudocode, a description of how your code solves the problem, and a brief argument of its correctness.

**Problem 1: Jarvis March (Gift Wrapping Algorithm)** For this question you will need to research the *Jarvis March* convex hull algorithm. Be sure to cite your sources (ACM or IEEE formatted is preferred).

**(a) [10 points]** Give pseudocode describing the Jarvis March algorithm, a brief description of how it works, and explain its best and worst case efficiency.

**(b) [5] points** Give an example input on which Jarvis March will perform significantly better than Graham's scan and explain why it will perform better.

**(c) [5] points** Give an example input on which Graham's Scan will perform significantly better than Jarvis March and explain why it will perform better.

**Problem 2: Find the Missing Number** You are given a list of $n-1$ integers $A$, in the range of 1 to $n$. There are no duplicates in the list. One of the integers is missing. (Feel free to assume that $n = 2^m$ for some integer $m$)

**(a) [5 points]** Give an efficient algorithm for finding the missing number, show its complexity, and argue its correctness. (You should try for $O(n)$-time and $O(1)$-space, less efficient solutions will still get partial credit)

**(b) [10 points]** For this question you are not allowed to access an entire integer with a single operation. The elements of the list are represented in binary, and the only operation you can use to access them is GetBinaryDigit(A[i],j) which returns the $j$th bit of element $A[i]$ which runs in constant time. Give an efficient algorithm for finding the missing number under these constraints, show its complexity, and argue its correctness. (You should try for $O(n)$-time and $O(logn)$-space, less efficient solutions will still get partial credit)

Example: If we run GetBinaryDigit($A[i]$,$j$) with $A[i] = 29$ and $j = 2$, it would return a 0 since $29 = 11101$.