Due on February 12th, 2018
30 points total

General Directions: If you are asked to provide an algorithm, you should clearly define each step of the procedure, establish its correctness, and then analyze its overall running time (for this assignment, this means arguing why your algorithm achieves the target running time specified by the question). There is no need to write pseudo-code; an unambiguous description of your algorithm in plain text will suffice.

All the answers must be typed, preferably using LaTeX. If you are unfamiliar with LaTeX, you are strongly encouraged to learn it. However, answers typed in other text processing software and properly converted to a pdf file will also be accepted. Before submitting the pdf file, please make sure that it can be opened using any standard pdf reader (such as Acrobat Reader) and your entire answer is readable. **Handwritten answers or pdf files that cannot be opened will not be graded and will not receive any credit.**

Finally, please read the detailed collaboration policy given on the course website. You are **not** allowed to discuss homework problems in groups of more than 3 students. **Failure to adhere to these guidelines will be promptly reported to the relevant authority without exception.**

Problem 1 (15 points)
Your friend Bob Bitfiddler has now come to you with a different problem. He is looking to open convenience stores in the small town of Linesville, NC. Linesville has just a single east-west street called Straight Street, along which there are n homes in total. Having surveyed the residents of Linesville, Bob has come to know that they are willing to walk at most one mile to go to a convenience store. Can you help Bob set up a minimum number of stores such that every resident has one within walking distance? The locations of the houses are given to you from east to west, and your algorithm should run in $O(n)$ time.

Problem 2 (15 points)
You are already planning a party for spring break! You have a large a group of friends, but not all of your friends are friends with each other (and you know of all the friendships between your friends). You wish to invite as many friends as possible to the party, but you do not want anyone to be lonely. So, you decide that you will select a subset of friends such that each invitee has at least 5 friends among the other invitees. Define an $O(m)$-time algorithm that finds such a subset of maximum size, where m is the total number of friendships among your friends.