Keys to a Great Problem Set

1 Content

1.1 Description/Explanation

1. Every solution should consist of pseudo-code accompanied by a clear explanation given in full sentences. It should be written in enough detail that someone outside of the class (but with CS background) could completely understand why it works and how to implement it. The reader should not have to make any assumptions about your solution or guesses about what you meant.

2. Depending on the problem, the pseudo-code may be a high level bullet list (e.g. sort, then traverse the array and look for duplicates). Other times, the pseudo-code will look more like code. It depends on the problem and what sort of primitives it uses. Some problems youll be able to use algorithms and data structures from class as building blocks. Others wont.

3. You can use any of the algorithms discussed in class without explaining. For e.g. you could say Build a max heap. Or insert element into heap. However, you still need to write pseudocode, just that the primitives in your pseudocode can be higher-level primitives that we learnt in class.

4. Since many people are unfamiliar with pseudo-code, we will not take off for small mistakes or style in the beginning, but we may mark them to help you. Pseudo-code is an important skill so by the end of the semester, everyone should be pros at this.

5. Do not explain things that are obvious. For e.g. its sufficient to say sort the array using an O(n lg n) sort. You do not need to write any more details than this.

6. Keep your explanation structured. Avoid writing an essay.

7. The ideal is to write it so clearly that someone who knows nothing about algorithms would understand it.

8. Assume the grader does not know the solution and that they actually need it to be explained. How you write it matters.

9. Consider all cases in your description. (only if applicable): Base-cases; Edge-cases; Is N even or odd; Etc.. Often it can be as short as saying Well assume for simplicity that n is even. The case n odd can be handled similarly.

10. Images and examples are great but they are only supplemental and do not count as a description. However, please include if it will help.

1.2 Justification

Describe why your solution is correct, that is, why it is doing what its supposed to be doing. Sometimes this will be obvious, sometimes not so much. Even if you think its obvious, write a short sentence arguing very briefly why the result is correct. Correctness should always be on your mind, and on your paper! (Why does it work? Does it always work? Any special cases that you need to be careful?)

1.3 Runtime Analysis

After you describe your algorithm you must analyze it. What is its worst-case runtime? This is very important.

2 Format

1. This is a graded assignment so it should be neat, organized, and formal. Always write in full/complete sentences and do not abbreviate words.
2. Leave plenty of space between problems to make it easy to read and so that we can write comments.
3. Write your answer top down. Do not write on the side of the page, do not write side to side. Make it easy to read.
4. It does not need to be an essay but often it takes a full paragraph explanation.
5. Your language should be formal.
6. Do not leave things crossed out. Either erase it completely or use a new piece of paper.
7. You will lose points for messiness.
8. Make your solution obvious. We should not have to search for your solutions; they should be clearly labeled or marked.
9. Write legibly. If you have particularly difficult to read handwriting, consider typing or writing especially clearly so I can understand. I shouldnt be guessing what you meant.
10. Be concise. Again, dont write an essay.
11. Keep the problems in order.

3 Attach the in-lab section

Answers are not graded but it is important that you attempt every problem.