INF-1100: Introduction to programming and computer behavior

Assignment 3

Kasper Hansen

October 8th, 2021

# Introduction

The introduction should motivate the document you are writing and put it into context. You should make the document’s purpose clear and describe what the reader will learn from reading it.

For documenting assignments, it can be beneficial to describe your understanding of the tasks you were asked to solve. It could be that your understanding is slightly different that the person reading it. Describe assumptions, shortcuts, or simplifications you made; or particular difficulties or surprises you encountered along the way.

Be short, interesting and specific, rather than long, boring, and wage. Punctuate and move on.

# Technical Background

This section should describe topics and concepts from the curriculum that are directly relevant to this assignment. The purpose of a technical background is twofold. First, in ensures that both you and the reader have the same knowledge and mental framework. Second, it convinces the reader that you know your stuff. For documenting assignments, the latter is probably most important.

Be specific and use the terms and concepts you have learned. You only need to sample a few topics. To be convincing, you not only need to show that you know stuff, but also that you can distinguish the important bits from the unimportant ones. For instance, if your assignment is about sorting integer arrays, writing passionately about impressionist painting techniques will not impress.

# Design and Implementation

Between the technical background section and the conclusions section, you are to write one or more section that describes what you have worked on and your results. If this section grows large (e.g., more than 1 pages) or cover clearly separate topics, we recommend dividing the text further up into multiple sections or subsections. Think about what you are trying to convey before committing to a specific structure.

Do not be afraid to restructure.

Do not use structuring elements that clearly does not belong, like the two subsections below.

## What to write about?

The most important element of writing is to figure out what to write about. If you spent time on the introduction, you probably have some idea by now. All you need to do is to put those thoughts into word. That is the difficult bit. Below, you will find some ideas what to include:

* Program components and how they relate
* Data structures
* Algorithms
* Theorems and theory
* Choices and tradeoffs
* Performance

The list is not exhaustive.

## Common themes

In addition to introductions and technical background sections, there are several themes that we often use to structure a report. These include:

* Design. An abstract description of your program. How it is organized, architecture, etc.
* Implementation. Code level details of your program, but generally avoid plain code.
* Discussions. Describes the choices you made and their implications.
* Evaluation. Proves that your system works (as intended).

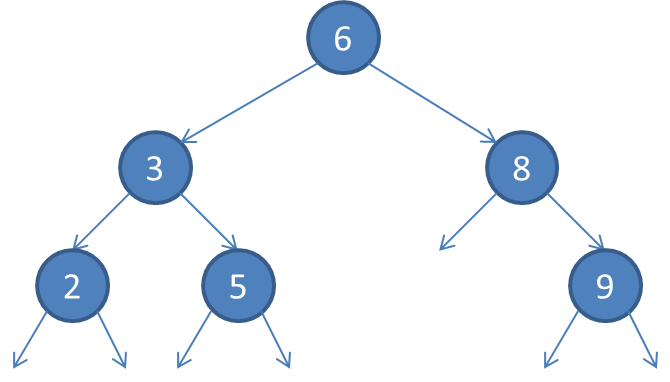
Some suggest having separate sections for each of these. However, as it is often difficult to separate one from another, do not force yourself to such structures. For instance, in this template we have combined design and implementation (Section 3) and have a separate evaluation section (Section 4). This might fit well for documenting a program where you also want to present performance evaluation. Discussions could be made part of both evaluations or design and implementation, where they are relevant.

## Elements

In addition to the main narrative (text), your documents may contain other artifacts. These include

* Figures and tables
* Algorithms
* Formulas
* Citations

Figures and table can be floated in the text. If so, remember to refer to them in the text, like we do here for Figure 1. Such elements should be thoroughly explained in the narrative.

**Figure 1**: A binary search tree

# Evaluation (optional)

If you designed and implemented a large and complex computer program, you probably also spent many pages describing what you did. You now need to convince the reader that what you did make sense and that your program behaves as it supposed to. That might require equally many pages and separating this out in its own section might be help.

Convincing the reader that your program works as it supposed to might require equally many pages.

# Discussion (optional)

You should try to actively discuss your work along the way. Sometimes, though, you find that you have something important to discuss that would interrupt the narrative elsewhere or require significant space and attention. In these cases, a separate discussion section can be warranted. Usually, you want to have these in the end.

# Conclusion

Sum up by restating the problem, your work, and your findings. What did you learn? Did you complete the task? There should be no surprises here.

Acknowledgement (optional)

Describe whom you worked with and on what parts others have contributed. Be careful to collaborate to closely on assignment that are not group works.

References

|  |  |
| --- | --- |
| [1] | R. Sedgewick, Algorithms in C - parts 1-4, vol. 9, Addison-Wesley Publishing Company, 1998, pp. 223-252. |