“Working Layout Document” contains the ideal layout for the final page. The music should be placed in the bottom margin of each page. The margins that will be used for pages are .75in on the top and 1.25in on the bottom to accommodate for the music. This page has the sample margins, and so does the Working Layout Document. Since the margins will add up to two inches on the top and bottom, individual writing pieces do not need to have the updated margins; though individual pages 1..48 MUST necessarily have these margins so that they can be printed correctly. Remember that Working Layout Document is NOT the final printed document. The individual page pdfs will be concatenated and modified with python into final.pdf. THAT’S what will be printed (preferably on a printer that doesn’t mess with the margins; see below paragraph).

I’m having trouble making the printer print with correct margins (Left margin is 2.54cm, right margin is 3.74cm?! seems like it’s being resized). If I use a proper printer to print the final thing, this shouldn’t be a problem... but there’s no real guarantee about that. The PDF looks right, though, so I’m going to dismiss worries right now.

Do try to keep Working Layout Document up-to-date with the text of content if it is changed, so that page flow and line count can be accurately measured. Try to keep content out of Working Layout Document if it is not yet finalised.

PDFs of each individual page should be uploaded to GitHub. PDFs MUST be PRINTED using the “Save to PDF” feature in Chrome, NOT exported using the “Download As” feature. Otherwise line spacing gets all screwy. Kerning appears fine in the printed PDFs. The uploaded PDFs should not have page numbers.

A python script will be written using pyPDF, its successor, or some other tool that will add page numbers (ToC is page #0, everything after that is incremented by 1. Thinking of putting page numbers in hexadecimal). The script will also render and add the music lines, likely written in MusicXML, to the bottom of the pages. We’ll probably have a directory of files named [1..48].xml, and those will be imported with music21 or something and exported to an image. The image will be placed in the bottom margin.

Update: Script written. Hooray! It works great. Just plop pages with filenames 0.pdf, 1.pdf, etc. in page-pdf, and musicxml files with filenames 0.xml, 1.xml, etc. into music-xml. Run pdf-scripts/do-page-generate.py, ensuring that ly-to-append.txt is also in the directory, for each page (setting the pageNum variable to the correct number each time). This can be looped, of course. Sit back and have pride in oneself. The useful output will be in pdf-output, and all the other folders (music-ly, music-pdf, pagenum-pdf) contain in-between files that can be removed.

Cambria is a pretty font, and is still my favourite. The only other font I would consider is Computer Modern, but I’m not sure how character spacing would work out. The font size might need to be changed to accommodate it, which would be ugly. I like Cambria more anyway. Text rendering is justified. In theory, this will not change the line count; though sometimes Docs is funky. The font is small enough that justification should not look too funky. When changing the alignment of an entire

document, make sure to keep the right-aligned things right aligned, like two-column tables used for alignment. Droid Sans Mono 10pt is used for code. Left alignment should also be applied for all code.

Stylised (directional) quotation marks, single and double, should be used throughout most of the work. As a general rule, if you're writing in Cambria, you *should* be using stylised quotes. If you're writing in Droid Sans Mono, take great care to ensure that straight quotes, single and double, are being used throughout the code. This is going to need attention. SQL should always use single quotes (apostrophes) around field/table names: do NOT use double quotes or backticks.

Hyperlinks should all be removed. Since I'm printing black-and-white, they'll look awkward anyway. Just write them as normal, un-underlined URLs.

If you see red highlights anywhere, see if you can eradicate them. I used outlines because I was unsure of the information at the time (perhaps it was a page number), or because I needed to make a note about page layout. If you can be confident that the red highlighted text will no longer be necessary, make the change.

The "Contents" page is the important contents page. Working Layout Document need not always reflect the actual contents *on that page*, though it should always reflect the actual contents in the actual content so that the layout can be established. It is imperative that the contents are only one page long, because otherwise it will look stupid. If two columns are necessary, do *not* hesitate to use two columns (all the way down one, then all the way down the other).

Once all of the individual page PDFs have been processed, they should be concatenated into one "final.pdf" which will be accessible in a minimal JS pdf viewer at portfolio/index.html. I have not yet researched which JS viewer I'd like to use, but they certainly exist. If all else fails, pdf.js will save the day. If I find it completely necessary, I can turn the final.pdf into individual HTML document pages or something, and write a quick JS page swapper. The online version is mostly an afterthought, because that's not what I'm turning in as my final.

~~MP3s/OGGs/whatever should be accessible IN the pdf by clicking on the music line. I think that's doable, because it should just be a simple hyperlink.~~ This is certainly possible, but messing around MORE with the pdfs is not a good idea. It's hard enough as it is. This will be accessible in the webviewer somehow instead.

It is OKAY to modify document text/metadata to make the document look nice. If justification looks awkward at some point, feel free to add in words to pad the extra space. As long as it sounds natural, it's fine. Remember that WYSIWYG on the Google Docs pages, so you're really formatting specific page layout at this point.

When right-aligning text on the same line as left-aligned text, just use tabs to do the right-align. Tables have problems in Docs.