**Specification for the Music Macro Language to MIDI Project**

**Overview**

The music macro language (MML) is a music description language that has been used in one form or another since 19xx. There has never been any official specification, so each version and implementation varies slightly, and over the years the language has improved and evolved. My project is to write two programs that can be used in combination to convert multiple MML text files into one multi track MIDI file.

The first program used to do this is the actual converter that reads in a single MML file and outputs a single-track MIDI file. This is done multiple times to produce multiple files representing each track of the final MIDI file. The second program combines these single-track MIDI files into a single multi track MIDI file.

The advantages of having two programs are that each track can be listened to before combining them, and instead of one complicated program being written, two simple programs can be written which will likely be less complicated together than the one combined program.

**Programming Language Choice and Aids Used**

I have decided that the C programming language is the most appropriate choice for me to do this project. This is because I am most comfortable with writing C than any other language, and it has a plethora of compiler compiler software available, which is particularly useful for the challenges presented by the project.

I will use two compiler compilers to help me write both programs: the first is lex, a lexical analyser generator that takes a language specifying file and outputs C code that can scan it. The second is yacc – this is a parser generator that takes it’s own language specifying file and, again, generates C code that can read it. Lex and yacc are very often used together and perform best in this configuration.

I’ve decided to not use any libraries for this project aside from the standard C libraries, as I feel like this will make the completion of the project a more valuable and informative experience. One effect of not using libraries is that I will have to write my own code to write a MIDI file.

To compile my software I will use the GNU Compiler Collection (GCC) called by makefiles. This is a method I have used many times before, and is a standard way to keep a handle on the minutiae of compilation, such as optimisation levels.

As the project is quite complicated, and will likely have a large source code, version control will be essential. For this, I will use git and github to allow me to work on the project from multiple systems. This is another standard technique this is widely used.

**Usage and Example**

Both programs written for this project will be called via the terminal. The first program, that turns the MML files to MIDI files, will be called as follows:

mmltomidi [-o output\_path] [file]

The “-o” switch and output file name is optional: if it is not included then the output of the program will be stored in a file called “output.midi” in the folder in which the program is run.

The second program, that turns multiple single track MIDI files into one multi-track MIDI file, will be called as follows:

catmidi [-o output\_path] [file …]

Again, the “-o” switch and output file name are optional, and if not included, the output of the program will be stored in a file called “output.midi” in the current working directory. The number of input files named at the end of the command is variable, but there must be at least one entered.