Effectiveness of Solution and Improvements:

The testing of my program has shown that it can identify a song with a success rate of about 86%, in an average time of 7 seconds. Since I initially aimed to identify a song in under 15 seconds, I feel that the program does function effectively.

One key aspect of the power of the search algorithm I used is that it looks for *two* matching notes between the database fingerprint and the query, separated by a certain time interval, simultaneously. This greatly reduces the number of false positives returned from the search. If we were just looking for one matching note, this would give a huge number of results from the search, many of which would be incorrect matches. Clearly then, we could improve the search time even more by looking for more matching notes between songs, at a time. For example, if I used four anchor points per target zone instead of just one, the search would be looking for five notes, each separated by five specific lengths of time. This would dramatically reduce the number of results required to then filter.

Another method of improving the search time could be to utilise multiple databases. Currently I am only using one and hence all the processing must be done serially on that one database. However, if I had N databases, where each one held 1/N of the total number of songs, then each one could perform a search and establish the closest match individually. This would give N potential closest matches. We then filter these N separately and choose the closest of them to the song. This could potentially speed up the search by a factor of N.

The final point I will touch upon for improvements is the many constants and thresholds that I've put into the algorithm at certain points. Here are the constants I've used and the potential effects of changing them:

- For the Fourier Transform, I set the window size to be 1024 samples. This means that the frequency resolution of the output is about 43Hz, and the time resolution is 23 milliseconds. Increasing the window size would mean a better frequency resolution therefore two frequencies closer together could be distinguished, but the time resolution would decrease, i.e. two notes closer together might be merged into one. Decreasing the window size would have exactly the opposite effect. For the sake of my project I think anywhere from 1024-4096 sample for the window size would yield very similar results hence I feel that there is little to improve here.
- I down sampled the audio by a factor of 4, from 44100Hz to 11025Hz. If I down sampled even more then the processing time would increase since there are less data points, but the accuracy would decrease. Since my user said accuracy was more important than speed, I thought down sampling by 4x was probably the best option.
- For locating the powerful frequencies I split each output from the Fourier Transform into 8 logarithmic bands. Decreasing the number of bands would provide fewer high frequency points in the fingerprint hence lower the chance of a match and also lower the accuracy of the search. It would however increase the search speed.
- Also while locating powerful frequencies, having taken the mean amplitude of all the points picked out by the logarithmic bands, I compared each point's amplitude against this average multiplied by a coefficient of 0.3. This value came about by experimentally changing it to different numbers and seeing what happened. I settled on 0.3 because it gave a good range of points between 0-5000Hz. Decreasing this

value would increase the noise robustness however, because there would be more points making it through to the final fingerprint.

- I've already spoken about the number of anchor points per target zone previously, but increasing this number would definitely improve search time.
- When it comes to the final searching, I accept the song if it has more than 50 matching target zones, and then of the songs that make it through this filter, I select the one which has the highest number of notes that match in the right order (and this value must also be greater than 39 else no result is returned). As before, I chose these values through my limited testing of certain possible values. This combination usually works to find a given song in a good amount of time. Lowering both of the values would decrease the search time because we could end the search the second a song passes a lower threshold, but it would make the accuracy lower, since a false song wouldn't need so many matches to be considered. Increasing the values would improve the noise robustness and the accuracy but increase the search time.