Gabrielle Bremer

CIS-2987

Anup Parajuli

# No-C-Notes

## Summary of the Project

No-C-Notes is an audio music transcription service created by Christina Cotruvo. Just like an audio book speaks the words from a book, No-C-Notes reads and spells out the sheet music for those with visual impairments.

The program takes XML files of music pieces and creates audio files that tell the user what notes to play in a piece. The main problem is sometimes the notes aren't written in the correct order in the XML files. To fix this, we use the default-x attribute and other elements in a note to assemble the audio track in the correct order.

## The Approach

My approach to solving this problem was to give each note a unique ID. This makes it easier for the sorting function to compare the unique ID of the previous note to the current one to determine which should be written first.

There are five elements that make up the ID for each note. The first is the measure number and I take this number and multiply it by 100000. Then I add the default-x number. That number is written as an attribute for each note within the XML. The third element is the note duration. For instance, this could be a whole, quarter, eighth, or sixteenth note and longer notes should be written first.

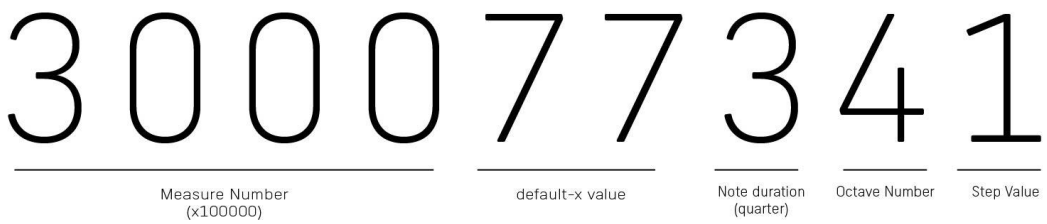
After the note duration comes the octave number. There are seven octaves and the lowest one should be written first. Lastly, the step value is appended to the end of the unique ID. This is the letter value in a note (low to high: C, D, E, F, G, A, B). Again, lowest tone should be written first.

Here is the dissection of a note in measure three of Happy Birthday.

## Happy Birthday - Measure Three

| UNIQUE ID | NOTES |
|---|---|
| 300000000 | Measure Three. |
| 300013342 | quarter-D four. |
| 300077341 | quarter-C four. |
| 300140345 | quarter-G four. |

Unique ID for quarter-C four




After unique ID was created for each note, it was added to a multidimensional array with all notes and their attributes from that measure. Then the array was sent to the sorting method I wrote and the sorted array was returned and the audio track is assembled.

Here is an example of where this sorting method was helpful:

## Sorting Example - Christmas in Killarney (M 15)

### BEFORE SORTING

BEFORE
Measure Fifteen.
Chord A minor.
eighth-E Four.
eighth-A four.
eighth-A four.
eighth-B four.
Chord A minor seventh.
eighth-C five.
quarter-A four.
eighth-B four.
with-half-E Four.

Notes are printed in order as laid out in the XML file

beforeSorting = { {1500343451, eighth, -, C five, .}
{1500414346, quarter, -, A four, .}
{1500530447, eighth, -, B four, .}
{1500343243, half, -, E four, .} }

### AFTER SORTING

AFTER
Measure Fifteen.
Chord A minor.
eighth-E Four.
eighth-A four.
eighth-A four.
eighth-B four.
Chord A minor seventh.
half-E Four.
with-eighth-C five.
with-quarter-A four.
with-eighth-B four.

Notes are printed by the unique ID order

afterSorting = { {1500343243, half, -, E four, .}
{1500343451, eighth, -, C five, .}
{1500414346, quarter, -, A four, .}
{1500530447, eighth, -, B four, .} }

After I finished sorting all notes correctly, there were a few smaller issues I needed to fix. I needed to code items for 6-4 time, add the mezzo-piano dynamic, and sort items in the voice map alphabetically so items are easier to find.

## Timeline & Hours

### Original timeline:

- Week 9: Recreate the loop that looks through each measure and sorts by the <default-x> value.
- Week 10: Continue to work on the loop and sorting each note by the default-x value.
- Week 11: Start working on sorting notations by the default-y value.
- Week 12: Continue to sort elements by default-y value, ensure this works with the default-x values.
- Week 13: Start to add rules and logic for elements that weren't included in the first version of the app.
- Week 14: Continue to add rules and logic for elements.
- Week 15: Run through many pieces of music to find bugs that may still be in the code.
- Week 16: Add sounds to the voice map.

Original estimation of how long it will take to complete each task

- Rewriting the loop and sorting all elements by default-x and default-y values: 20 hours
- Writing rules and logic for all new elements: 15 hours
- Other bug fixes: 10 hours
- Adding new elements to the database: 5 hours

**Total hours estimated: 50 hours**

## Tasks Completed & Dates:

**Actual hours spent: 57 hours**

**Week 10 Tasks**

- Take all music notes in one measure and put into an array
- Loop through the array and pull out the default-x number for each music note

In this first screenshot I'm taking all notes in a measure and putting them into a multidimensional array called orderList.

```
121
122         if (staff == 1 || staff == 99)
123         {
124             string stupidString = "";
125
126             foreach (string filename in items)
127             {
128                 if (!string.IsNullOrEmpty(filename))
129                 {
130                     tracklistStaff1.Add(filename);
131                 }
132
133                 if (!handed)
134                 {
135                     //&& Int32.TryParse(filename, out erasethis)
136                     if (!filename.Contains("Measure") && !filename.Contains("Hand"))
137                     {
138                         handed = true;
139                     }
140
141                 }
142
143                 if (!filename.Contains("Measure") && !filename.Contains("Hand") && !filename.Contains("1000") && !filename.Contains("250"))
144                 {
145                     noteOrRestRight = true;
146                 }
147                 stupidString += filename;
148
149                 //Adding note attributes to orderList
150                 orderList[x, y] = filename;
151                 y++;
152
153             }
154
155
156         }
157
```

Then I pulled out the default-x for each number and added it to the first spot in the array.

```
70
71          if (orderDefault != orderList[x, y])   ≤ 107ms elapsed
72          {                                      orderList  {string[100, 8]}
73              x++;                               [1, 7]   null
74              y = 0;                             [2, 0]   View  "58"
75              orderList[x, y] = o                [2, 1]   View  "eighth"
76              y++;                               [2, 2]   View  "250"
77          }                                      [2, 3]   View  "E Four"
78                                                 [2, 4]   View  "1000"
79                                                 [2, 5]   null
80          if (staff == -1)                       [2, 6]   null
81          {                                      [2, 7]   null
82              //MessageBox.Show("                [3, 0]   View  "129"
83              //trackStartSingle(                [3, 1]   View  "eighth"
84              foreach (string file               [3, 2]   View  "250"
85              {                                  [3, 3]   View  "A four"
86                  if (!string.IsN                [3, 4]   View  "1000"
87                  {                              [3, 5]   null
88                      tracklistSt
89                  }
90
```

**Week 11 Tasks**

- Create a sorting function to sorts all numbers by default-x from smallest to largest (5.5 hours)
    - Send each measure to the sorting function by staff – 1.5 hours
    - Create the sorting function – 3 hours
    - Add the items in the new list to the track list that gets printed out – 1 hour

In this screenshot I created the method that will sort all of the notes.

```
1 reference
public static string[,] SortList(string[,] orderList, int[] staffs)
{

    // Get the length of the first dimension (number of rows)
    int rows = orderList.GetLength(0);

    // Create a list of tuples to store the array values and their corresponding first number
    List<Tuple<string[], int>> tupleList = new List<Tuple<string[], int>>();

    // Iterate over the array and add each row as a tuple to the list
    for (int i = 0; i < rows; i++)
    {
        // Parse the first value in the row to an integer
        int firstNumber;
        if (int.TryParse(orderList[i, 0], out firstNumber))
        {
            // Add the row as a tuple to the list
            // Copy the row into a string array
            string[] rowArray = new string[orderList.GetLength(1)];
            for (int j = 0; j < orderList.GetLength(1); j++)
            {
                rowArray[j] = orderList[i, j];
            }

            // Add the row as a tuple to the list
            tupleList.Add(new Tuple<string[], int>(rowArray, firstNumber));
        }
    }

    // Sort the list by the first number
    tupleList.Sort((a, b) => a.Item2.CompareTo(b.Item2));

    // Create a new array to hold the sorted values
    string[,] sortedArray = new string[tupleList.Count, orderList.GetLength(1)];

    // Copy the sorted tuples into the new array
    for (int i = 0; i < tupleList.Count; i++)
    {
        for (int j = 0; j < orderList.GetLength(1); j++)
        {
            sortedArray[i, j] = tupleList[i].Item1[j];
        }
    }

    // Return the sorted array
    return sortedArray;
}
```

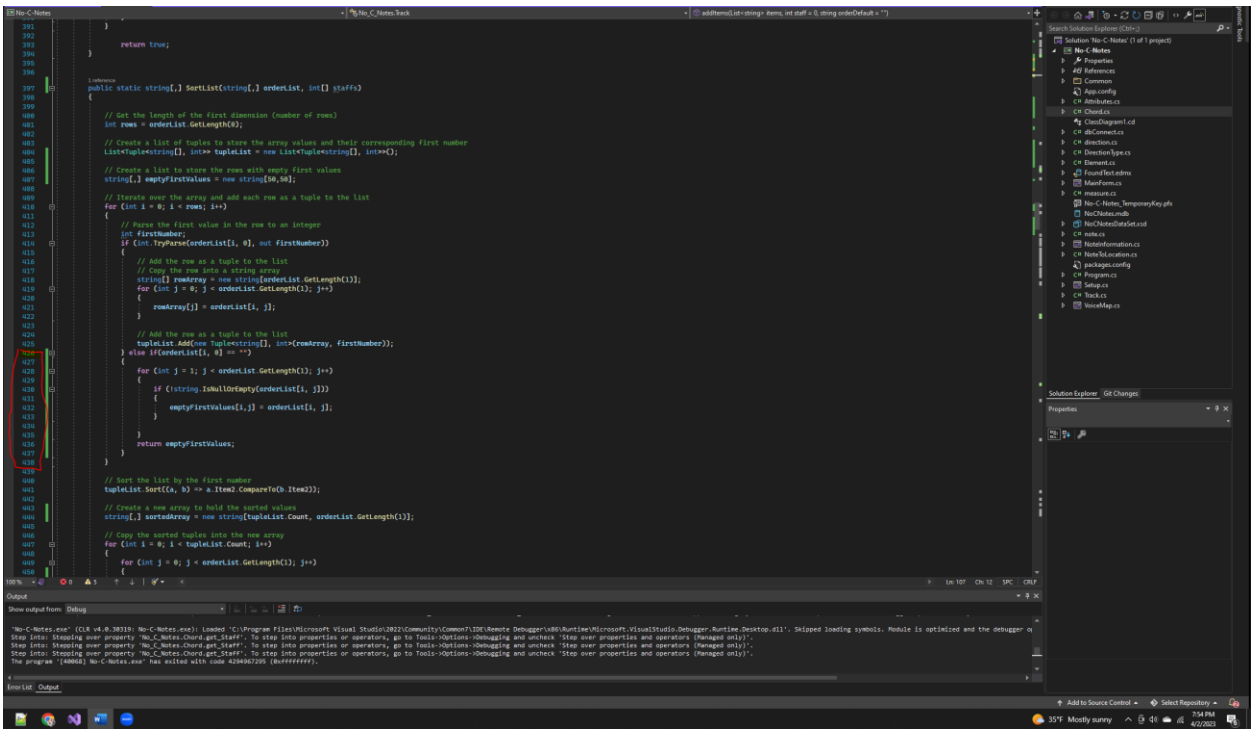
And here is where we loop through all of the notes and send the array to the sorting method shown above.

```
//Adding staff to an array with staff numbers
staffs[staffNum] = staff;
if (staffNum > 0)
{
    //If staff does not equal the previous staff
    if (staffs[staffNum - 1] != staffs[staffNum])
    {
        //send orderList and staffs to sorting function
        //Store sorted staff in new array
        string[,] newArr = SortList(orderList, staffs);


        if (staff == 1)
        {
            //Loop through each note in the sorted array and add to the tracklist
            foreach (string item in newArr)
            {
                if (!string.IsNullOrEmpty(item))
                {
                    tracklistStaff1.Add(item);
                }
            }
        }
        else if (staff == 2)
        {
            //Loop through each note in the sorted array and add to the tracklist
            foreach (string item in newArr)
            {
                if (!string.IsNullOrEmpty(item))
                {
                    tracklistStaff2.Add(item);
                }
            }
        }
        else if (staff == 99)
        {
            //Loop through each note in the sorted array and add to the tracklist
            foreach (string item in orderList)
            {
                if (!string.IsNullOrEmpty(item))
                {
                    tracklistStaff1.Add(item);
                }
            }
        }


        //Clean orderList for next hand
        Array.Clear(orderList, 0, orderList.Length);

        x = 0;
        y = 0;
    }
}
//incrementing staffNum for staffs array
staffNum++;
```

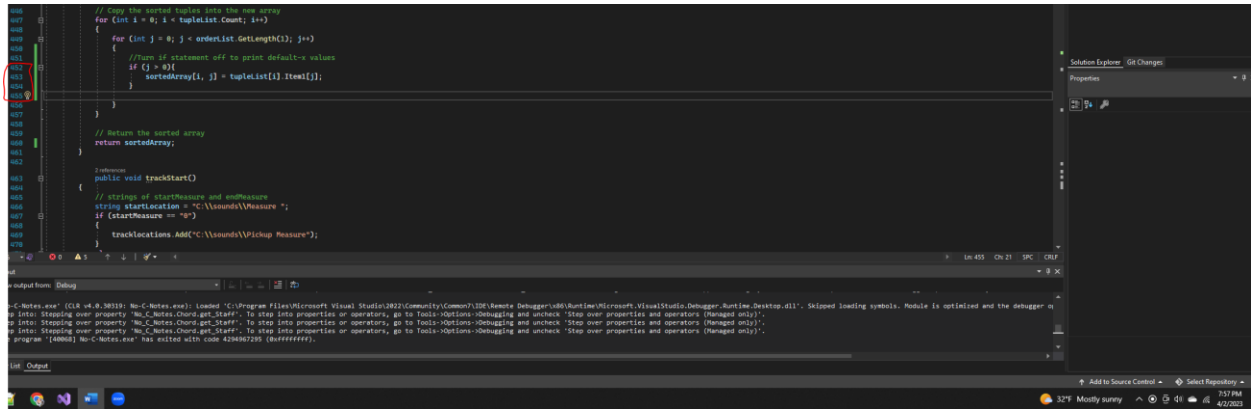
**Week 12 Tasks**

- Printing all notes in order by default-x number – 4.5 hours
    - Printing out measure number for both hands – 1hour
    - Printing out all notes for both hands – 3 hours
    - Stop printing the default-x number – 30 minutes
- Start working on getting chords to print in the correct place – 2 hours
- Add another measure to test – 30 minutes

Here’s the adjustment to the sorting function to accept entries that do not have a default-x number tied to it. For example, “Measure” doesn’t have a value tied to it so it is printed in the same place it was found.



Next, I took the default-x numbers out of the printout.

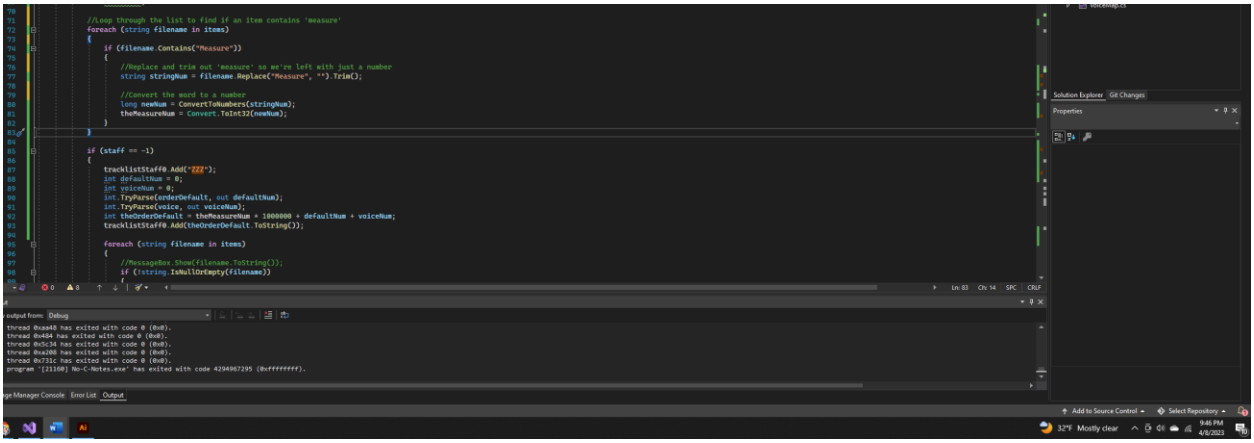


**Week 13 Tasks**

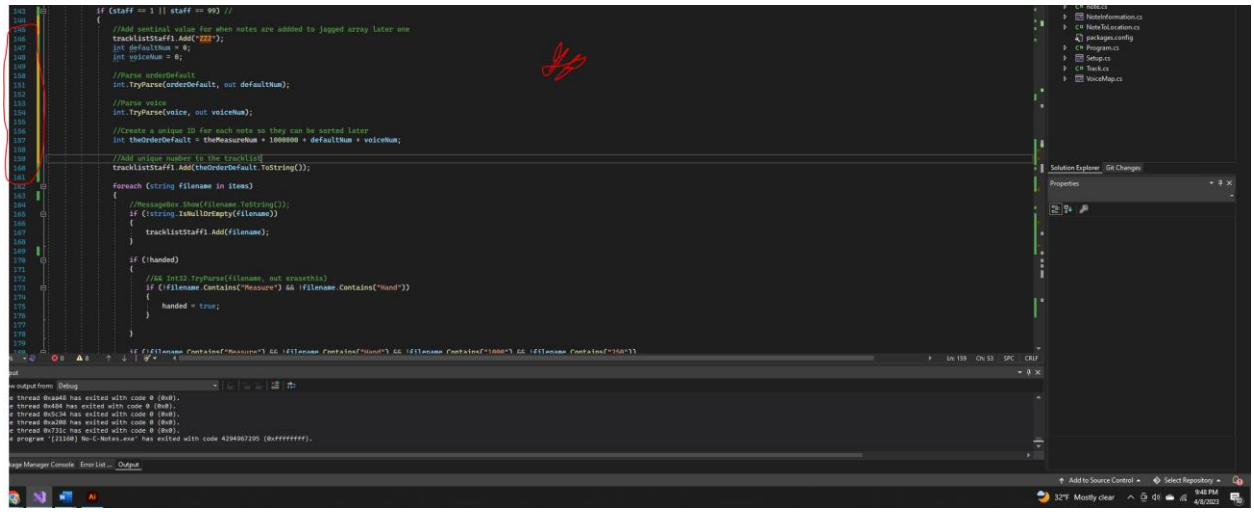
- Fix the last measure getting lost in the printout. – 4 hours
    - I had a meeting with a programmer where we tried to figure out why the last measure wasn't printing and then found out we needed my code to execute later in the program. – 2 hours
    - I spent the remaining two hours moving the code and putting the pieces back together. – 2 hours

- Fix the program so it sorts notes by measure  (Bonus) – 4 hours
    - First, I tried to mess with the sorting function. – 1.5 hours
- Test, test, test  - 1 hour
- Send to client to test  – 1 hour

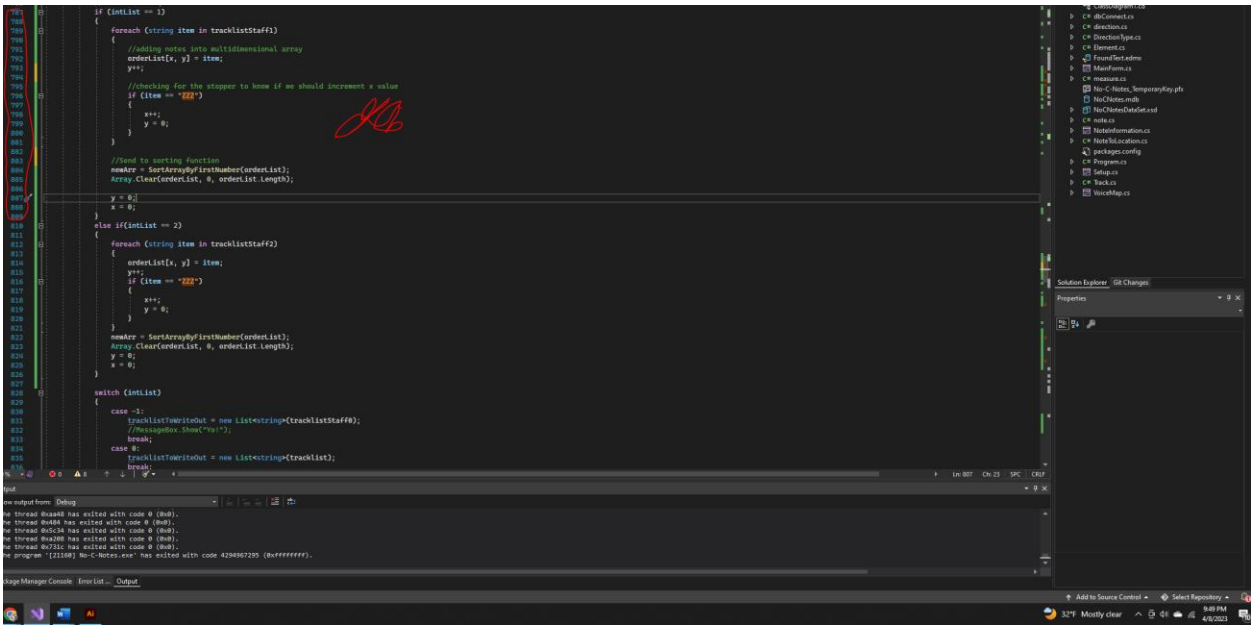
Here I looped through the list of attributes being passed into addItems and looked for one that contained “Measure” (an item would come in as ‘Measure One’). Then I would replace the word ‘Measure’ with empty quotes and trim the empty spaces. After, I would convert the word to a number to create the first part of the unique ID for each note.



Then, for each tracklist, I parsed the default-x and voice values to numbers. Then I multiplied the measure number times 100000 and added on the default-x and voice numbers to create a unique ID for each note. This was added to the tracklist.



Down where items are printed to the screen, I looped through the tracklist for each staff and added all of the attributes into a jagged array. Then I sent it to the sorting function.

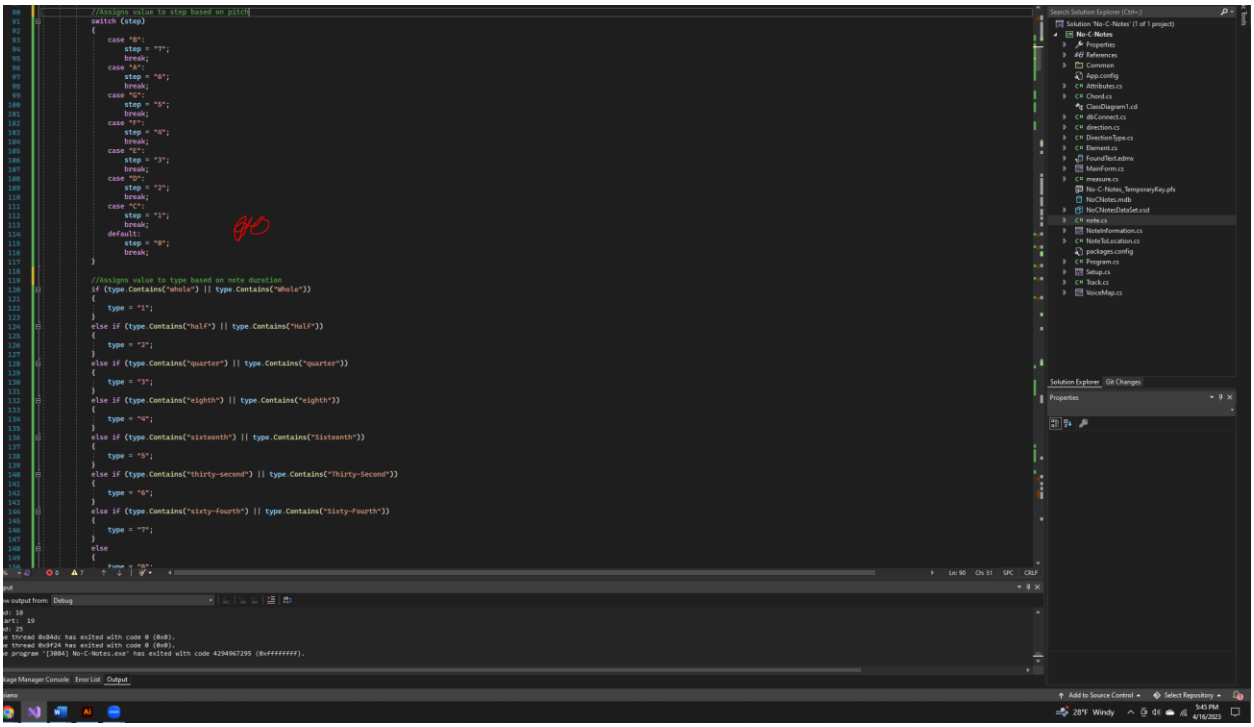


**Week 14 Tasks**

- Incorporate feedback from my client about the order of the notes – 5 hours, 45 minutes
    - Take out voice number and sort by note duration and pitch – 3 hours
    - Debug sorting errors – 2 hours, 15 minutes
    - Remove unique IDs from unfound items – 30 minutes
- Include ‘mezzo piano’ in the program – 1 hour
- Fix tied note errors – 1 hour
- Test with multiple pieces – 1 hour

During this week I started by reading the feedback I received from my client. There were a few instances where notes were still printing in the wrong order because the unique IDs were the same. My client’s solution was to sort by note duration (quarter note, whole note, etc.) if both default-x numbers are the same. If the note duration is the same, sort by pitch.

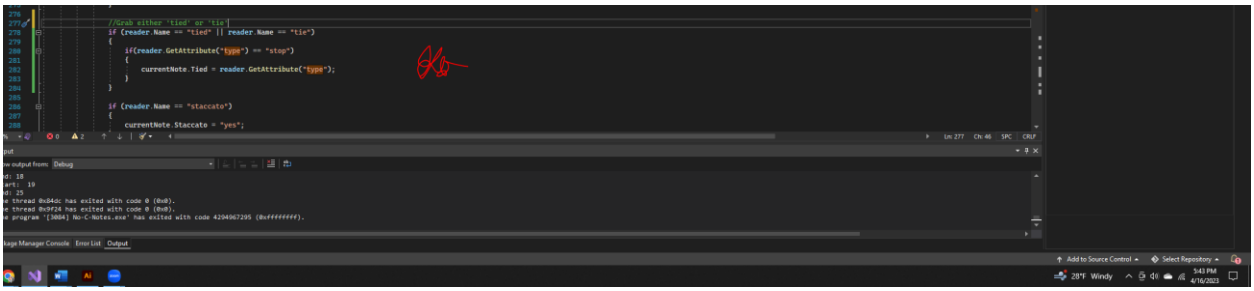
The first thing I did was pass the octave, step, and note duration into the method where items are added into the tracklist. Then I assigned values to those variables based on the advice from my client. She said longer notes should be printed first, so a whole note gets a value of 1. Then higher pitches should be printed first compared to lower pitches.



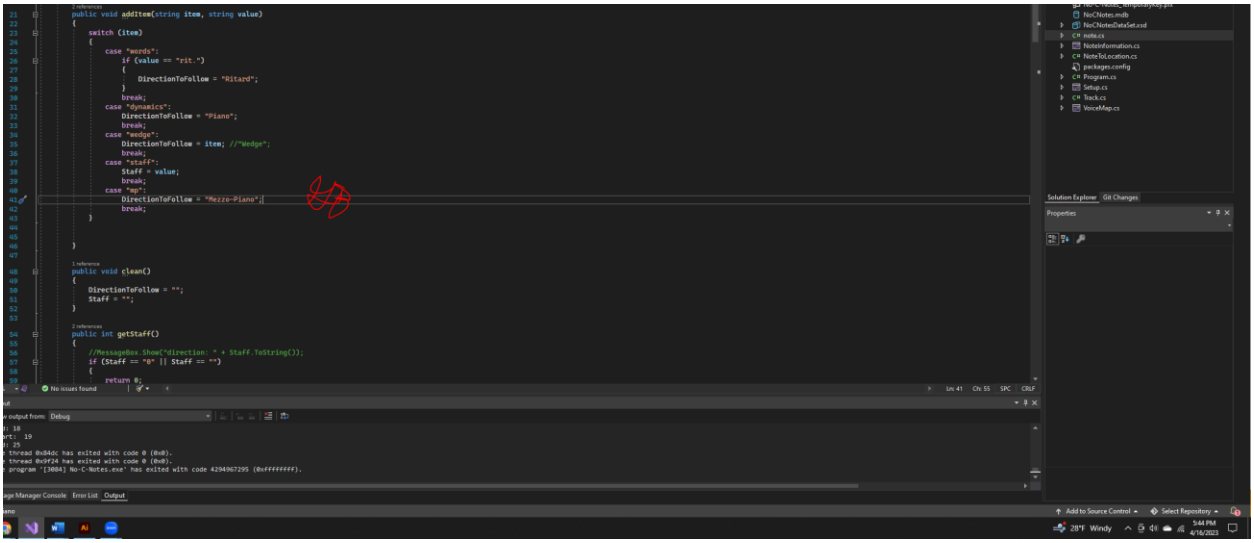
Next, I added it all together. I appended type, octave, and step to the unique ID to create a variable that will always be unique.

```
if (staff == 1 || staff == 99) //
{
    //Add sentinal value for when notes are addded to jagged array later one
    tracklistStaff1.Add("ZZZ");

    //If orderDefault is a decimal, take away the decimal
    if (orderDefault.Contains("."))
    {
        orderDefault = orderDefault.Replace(".", "");
    }

    //Parse orderDefault
    float.TryParse(orderDefault, out defaultNum);

    //Convert octave into a number - 'three' to 3
    long octaveNum = ConvertToNumbers(octave);

    //Convert octave back into a string - 3 to '3'
    octave = octaveNum.ToString();

    //Multiply measure number by one hundred thousand and add defaultNum
    float theOrderDefault = theMeasureNum * 100000 + defaultNum;

    //Append type, octave, and step to the orderDefault number and add to tracklist
    string order = theOrderDefault.ToString() + type + octave + step;
    tracklistStaff1.Add(order);

    foreach (string filename in items)
    {
        //MessageBox.Show(filename.ToString());
        if (!string.IsNullOrEmpty(filename))
        {
            tracklistStaff1.Add(filename);
        }

        if (!handed)
        {
            //&& Int32.TryParse(filename, out erasethis)
            if (!filename.Contains("Measure") && !filename.Contains("Hand"))
            {
                handed = true;
            }

        }

        if (!filename.Contains("Measure") && !filename.Contains("Hand") && !filename.Contains("1000") && !filename.Contains("250"))
        {
            noteOrRestRight = true;
        }

    }
}
```

After finishing the problems above, I moved on to fixing tied notes. Tied notes weren't printed in the correct places even though they were written in the XML. I found that the program was looking for the word "tied" when sometimes only "tie" was written in the XML. I modified the program below to also accept "tie."



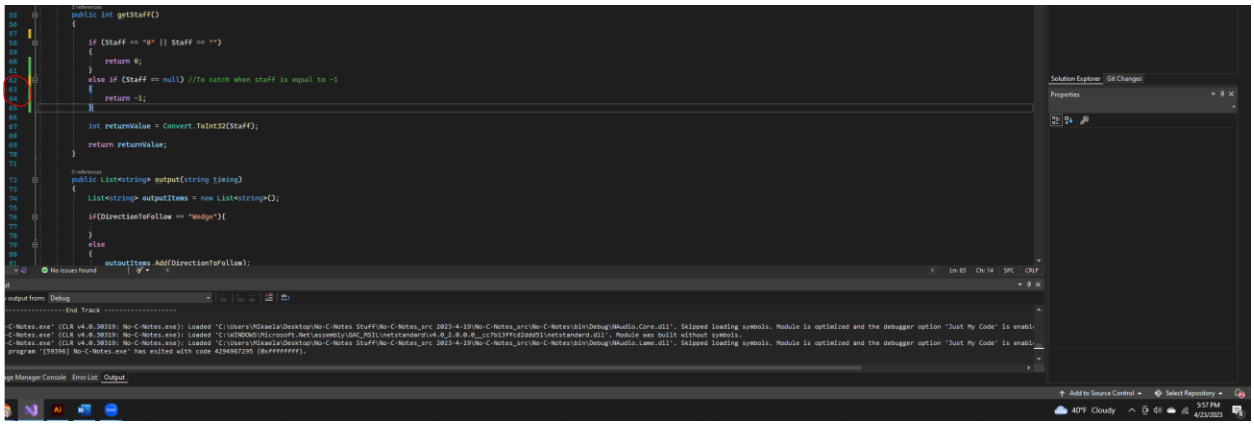
Lastly, there are different dynamics in music to tell the reader how loud to play. Piano is soft while mezzo-piano is half soft. The program didn't have a rule for mezzo-piano so it printed piano instead. I added a rule to this switch case to say if <mp/> is passed in, that means mezzo-piano should be written.



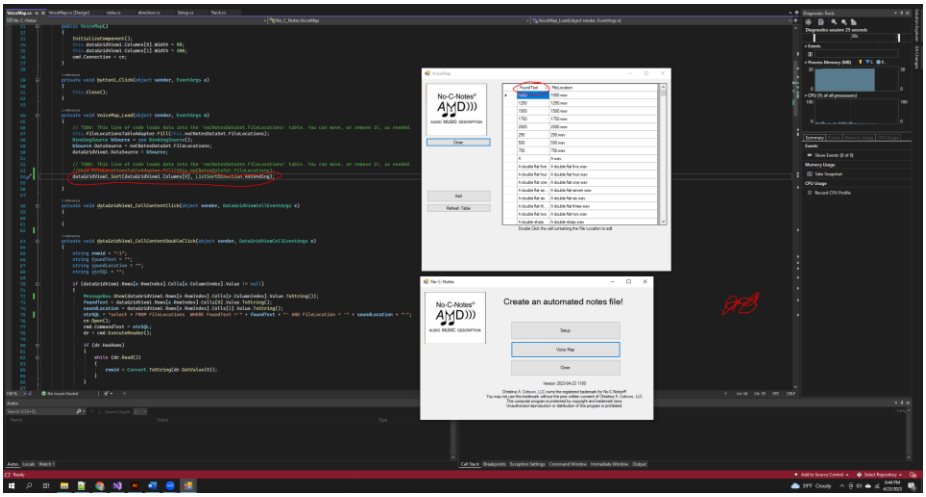
**Week 15 Tasks**

- Read feedback from client and make all octave tones lowercase – 1 hour
- Fix mezzo-piano issue – 1.5 hours
- Fix 'with' issue – 3 hours
- Changes to voice map 1.5 hours

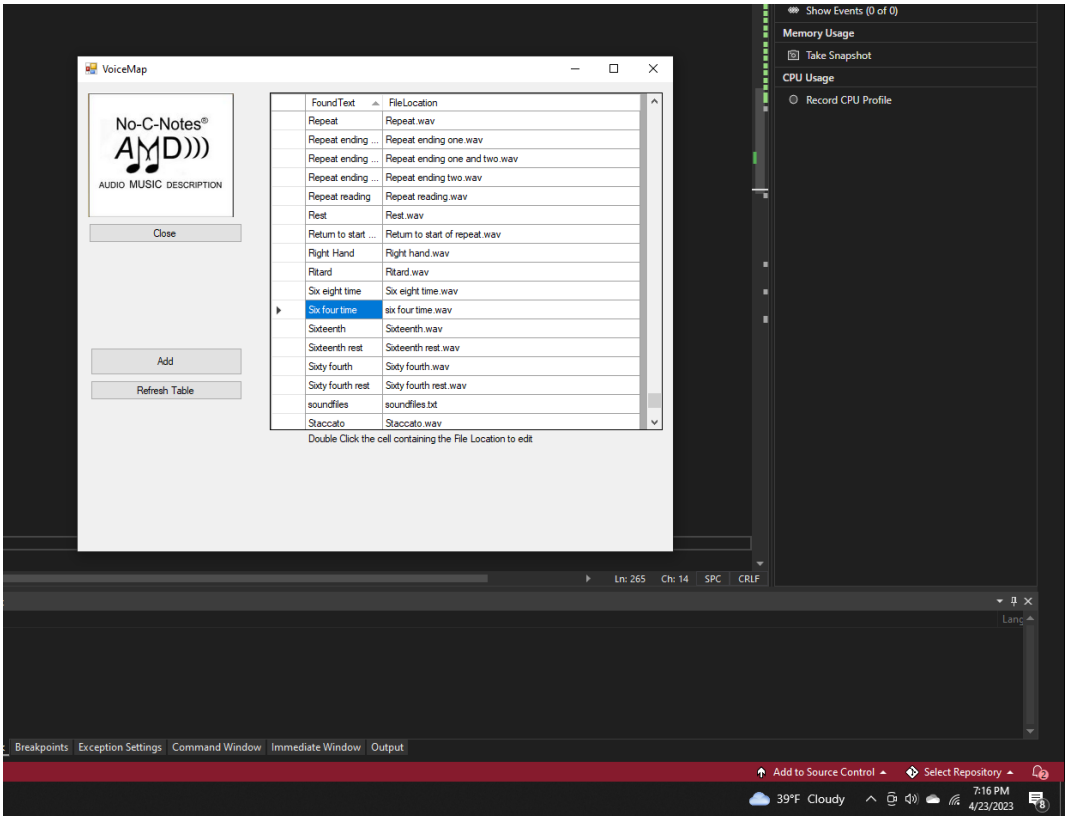
The first issue I tackled during week 15 was printing mezzo-piano in the right place. I realized that mezzo-piano was bring printed on the wrong staff. This piece of music didn't have a staff so it was supposed to be listed as -1 but instead was 0. I added three lines of code to this if statement that gets the staff to say if staff is equal to null then return -1 for the staff value.



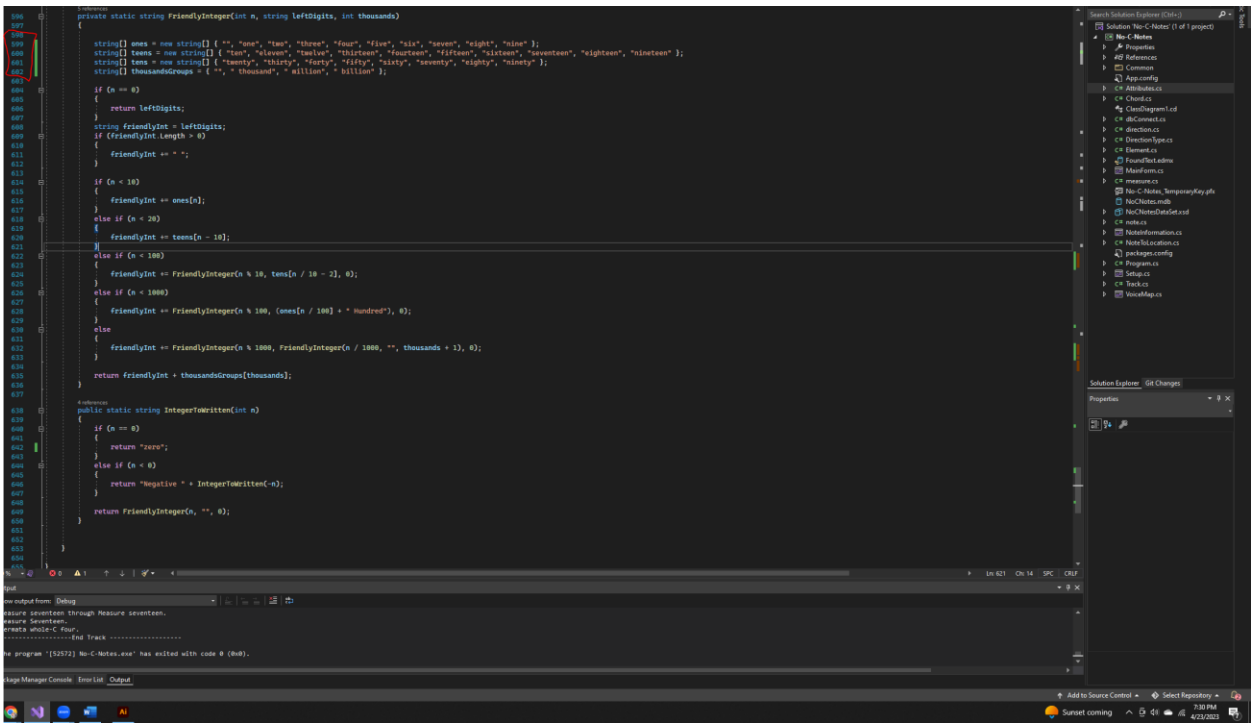
The client wanted all notes to be automatically sorted under the voice map's "Found Text" column for easier viewing. It was an easy line of code I added to line 54 of the screenshot below.



Next, I added six four time to the Voice Map because it wasn't in there already. However, six four time wouldn't show up when I sent a piece of music through the program.

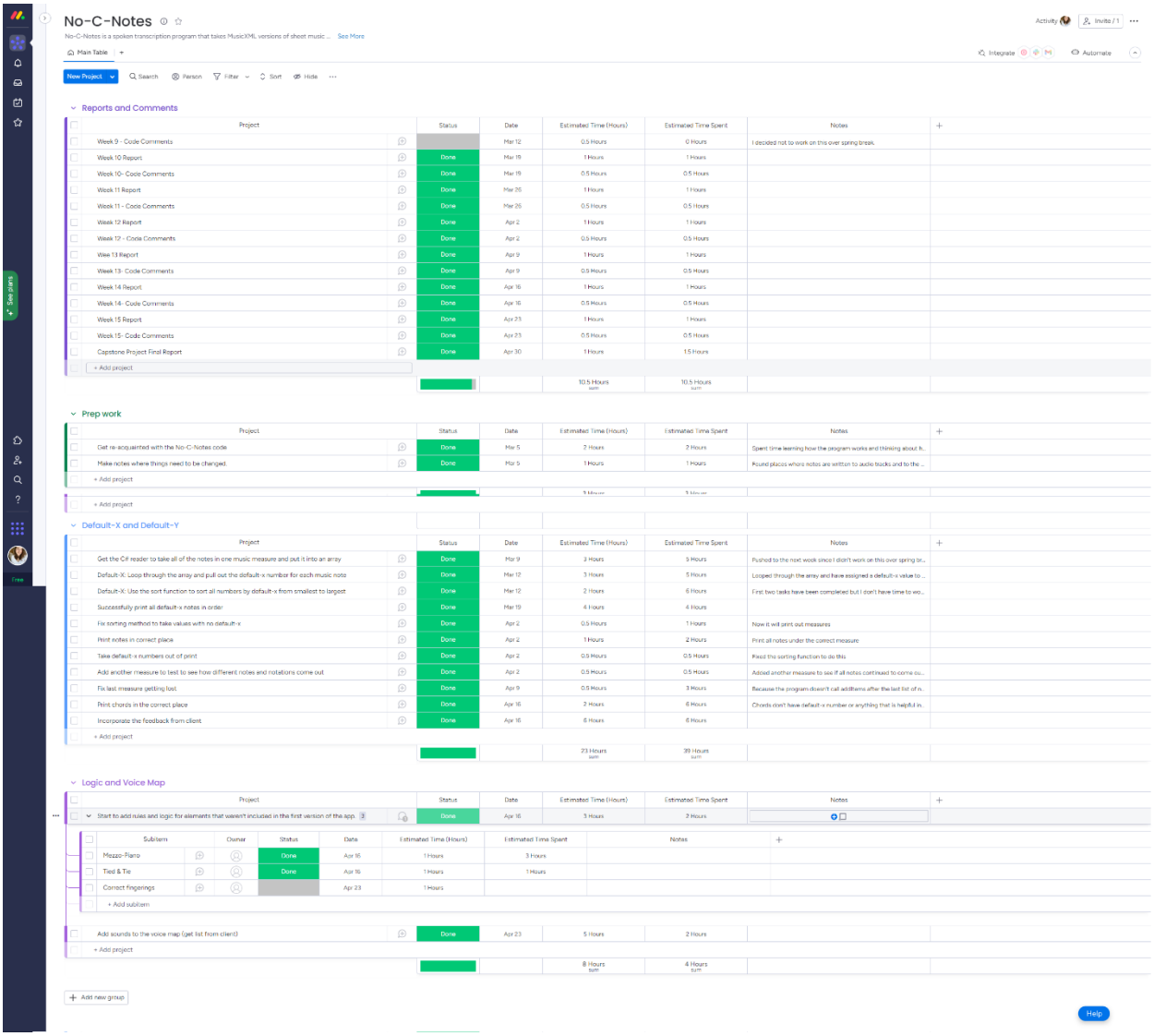



One of the last things the client requested I do was make all octave tones lower case. This was relatively easy, all I had to do was find where the notes were the numbers for the notes were bring printed and make them all lowercase. Now, instead of eighth-D Four, it's eighth-D four.



## PM Software

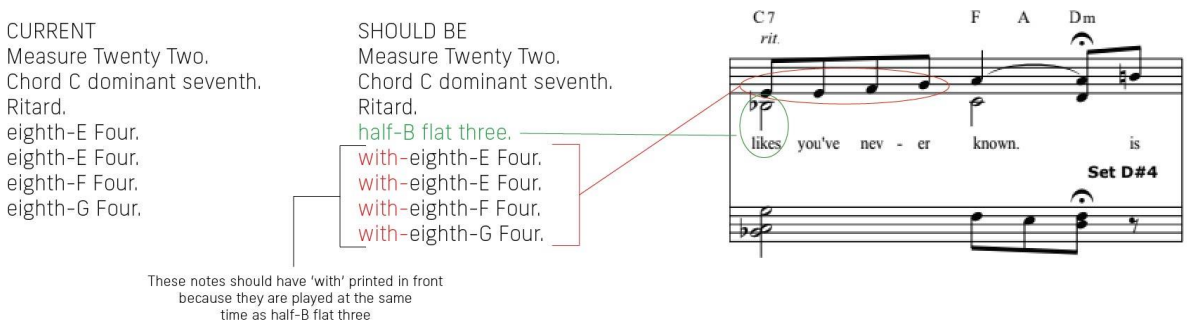
I used Monday.com to keep track of my project




## Tasks Not Completed

The only major bug I have not patched yet is determining when ‘with’ should be printed in a note. ‘With’ should be printed when two notes are played at the same time and the note with the longest duration should be printed first. Each shorter note after that should have ‘with’ in front of it. Here is an example:

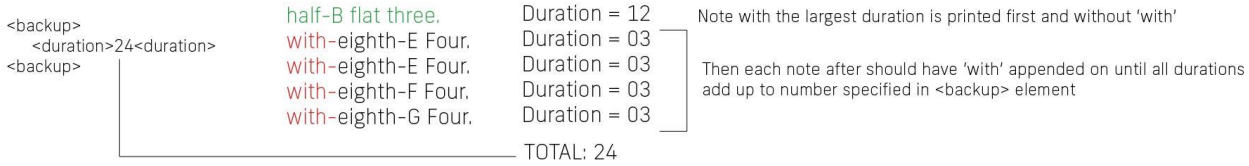
# ‘With’ Example - Christmas in Killarney (M 22)




## THE SOLUTION

Each time there is a chord (two or more notes played at the same time), a <backup> element is printed in the XML and has a number value. Every note after that <backup> element has a <duration> value tied to it.

### AN EXAMPLE




## Reflection

Overall, this was a successful project and I completed most of the items on my list. I’m happy I was able to help someone who is using their skills to help others.

Throughout this project, I learned that things don’t always go as planned so it is necessary to adapt quickly to set the project on the right course again. I also learned to overestimate my timelines on coding projects. I thought I was being generous by quoting 50 hours in the beginning. However, I hit roadblocks throughout the project and it took me longer than expected. 57 hours is probably a conservative guess, it may have easily taken almost 70 hours to get as far as I did.

Lastly, I am proud of myself for taking on this project when I knew it was going to be hard. Before this, I only had one semester of C# experience, and this is the biggest program I’ve ever worked with. While I struggled a lot, I was persistent and managed to complete most of the project.

## Contact Information

**Christina Cotruvo**

Email: christina.cotruvo@gmail.com