The Mathematics Behind the Music: Musical Enumerations, Asymmetric Rhythms, & Crab Canons

A Thesis Presented to

The Faculty of the Mathematics Program

California State University Channel Islands

In (Partial) Fulfillment

of the Requirements for the Degree

Masters of Science in Mathematics

by

Marissa Dahme

May 2012

Signature page for the Masters in Mathematics Thesis of Marissa Dahme

APPROVED FOR THE MATHEMATICS PROGRAM

Cynthia J. Wyels 5/23/12

Dr. Cynthia J. Wyels, Thesis Advisor Date

R A Roybal 5/22/12

Dr. Roger Roybal, Thesis Committee Date

APPROVED FOR THE UNIVERSITY

5-30-12

Dr. Gary A. Berg, AVP Extended University Date

Non-Exclusive Distribution License

In order for California State University Channel Islands (CSUCI) to reproduce, translate and distribute your submission worldwide through the CSUCI Institutional Repository, your agreement to the following terms is necessary. The author(s) retain any copyright currently on the item as well as the ability to submit the item to publishers or other repositories.

By signing and submitting this license, you (the author(s) or copyright owner) grants to CSUCI the nonexclusive right to reproduce, translate (as defined below), and/or distribute your submission (including the abstract) worldwide in print and electronic format and in any medium, including but not limited to audio or video.

You agree that CSUCI may, without changing the content, translate the submission to any medium or format for the purpose of preservation.

You also agree that CSUCI may keep more than one copy of this submission for purposes of security, backup and preservation.

You represent that the submission is your original work, and that you have the right to grant the rights contained in this license. You also represent that your submission does not, to the best of your knowledge, infringe upon anyone's copyright. You also represent and warrant that the submission contains no libelous or other unlawful matter and makes no improper invasion of the privacy of any other person.

If the submission contains material for which you do not hold copyright, you represent that you have obtained the unrestricted permission of the copyright owner to grant CSUCI the rights required by this license, and that such third party owned material is clearly identified and acknowledged within the text or content of the submission. You take full responsibility to obtain permission to use any material that is not your own. This permission must be granted to you before you sign this form.

IF THE SUBMISSION IS BASED UPON WORK THAT HAS BEEN SPONSORED OR SUPPORTED BY AN AGENCY OR ORGANIZATION OTHER THAN CSUCI, YOU REPRESENT THAT YOU HAVE FULFILLED ANY RIGHT OF REVIEW OR OTHER OBLIGATIONS REQUIRED BY SUCH CONTRACT OR AGREEMENT.

The CSUCI Institutional Repository will clearly identify your name(s) as the author(s) or owner(s) of the submission, and will not make any alteration, other than as allowed by this license, to your submission.

The Math Behind the Music
Title of Item

Masters in Mathematics Thesis
3 to 5 keywords or phrases to describe the item

Marissa Dahme
Author(s) Name (Print)

Marissa Dahme 5/23/12
Author(s) Signature Date

Table of Contents

Introduction

Both mathematics and music can be described based on a sequence of rules, boundaries, or axioms. Different areas of mathematics are built upon certain axioms with the most well-known being Euclid's series of axioms and the troublesome parallel postulate. Different areas of music are built upon certain rules or boundaries created, including specified harmonics, major and minor keys, and rhythmic and time signature restrictions; yet Rothstein [10] suggests that "neither axiomatic music nor axiomatic mathematics is a complete or satisfactory description of these arts." He mentions that both music and mathematics are not only generated by a certain set of rules but come into being through "more fundamentally mysterious processes." Although both mathematics and music can be argued to have more artistic substance than what can be described through a series of rules, there is enough structure and design in both that we can see why one tends to describe them using certain boundaries or axioms.

The overall goal of this thesis is to provide a glimpse into the inner life of music that can be seen through the use of mathematics. Chapter 1 provides the reader with some background knowledge needed including necessary definitions for intervals and transformations that will be used in subsequent chapters. The main focus of the thesis will be on musical canons, specifically rhythmic canons and crab canons. Chapter 2 is a summary of work by Hall and Klingsberg on counting the number of asymmetrical rhythms which is then applied to rhythmic canons. Chapter 3 includes a brief summary of musical canons, an analysis of Bach's "2 Canon *a 2*" from Musical Offering, as well as a definition for isomorphic crab canons essential for the work in Chapter 4. Chapter 4 gives two separate algorithms for creating crab canons suggesting that a certain mathematical structure can be used to compose musical crab canons. Chapter 5 provides

the reader with possibilities for further research. A brief musical background and prior knowledge of combinatorics techniques are not necessary, but of course helpful, in the reading of this thesis.

Chapter 1: Musical Enumerations, Interval Classes, and Transformations

In this chapter we will discuss important definitions and give examples of how to enumerate certain musical permutations and count isomorphisms related to music. Most of what is discussed in this chapter is based on the article by Julian Hook [6].

Section 1.1 Interval Class

Musicians refer to the twelve different musical tones, each a half step apart, using one letter for each tone (letters A – G). For example, the tones C and D are one half step apart where as the tones C and E are two half steps apart. We can use numbers to refer to the musical tones starting with C = 0, D♯ = 1, D = 2, and so on.

In music, notes are often played simultaneously either through multiple voices or on one instrument such as the piano. When analyzing a melody we will often refer to the intervals between two notes as part of our analysis. The interval between notes can be viewed as the distance in half steps between the notes played. We will begin with the most basic definition for determining the number of half steps between two notes and progress through a series of definitions for an interval in order to determine the most suitable definition for our analysis.

Definition 1.1: Define pitch interval to be the number of half-steps that separates one pitch from another. Then we can define the pitch interval between notes a and b to be $PI(a,b) = b - a$.[8]

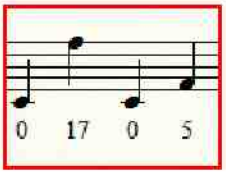
Example 1.1: Let a = C = 0 and let b = 17 (note F an octave above). Then PI(0,17) = 17 – 0 so the pitch interval is 17.

Using the pitch interval definition will only make two notes with the exact same difference of half-steps comparable. For example, the notes a = 3 and b = 20 have the same pitch interval, PI(3,20) = 17 as the notes in Example 1.1. What is more useful both musically and

mathematically is the pitch interval class. The pitch interval class is the pitch interval taking into account that there are only 12 unique musical tones.

Definition 1.2: Let the pitch interval class be $PIC(a,b) = b - a \pmod{12}$.[8]

Then the pitch interval class for Example 1.1 is PIC(0, 17) = 17 - 0 (mod 12) = 5. This means that $a = 0$ and $b = 17$ is equivalent to $a = 0$ and $b = 5$ and you can see that both values of b, 5 and 17 are both note F, just an octave apart.



Now using modular 12 arithmetic, we can refer to all possible notes using the numbers 0 through 11. Let all 12 tones be defined as in Table 1.1:

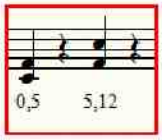
Table 1.1

Note	Value
C	0
C#	1
D	2
D#	3
E	4
F	5
F#	6
G	7
G#	8
A	9
A#	10
B	11

Using pitch interval class, we now have only twelve different intervals to consider, the intervals 0 through 11 (1 - 0 = 1, 2 – 0 = 2, 3 – 0 = 3, etc.). Note that having two voices play the note C =

0 and the note C = 0 will be in the same pitch interval class as having two voices play the note C = 0 and the note C = 12 since PIC(0,0) = 0 (mod 12) and PIC(0,12) = 0 (mod 12).

Example 1.2: When we disregard the order of the notes there are actually two different intervals to consider. For example take the interval created by playing the notes C and F. This provides us with two different pitch interval classes.



For the first interval we have PIC(0,5) = 5 - 0 (mod 12) = 5 and for the second interval we have PIC(5,12) = 12 - 5 (mod 12) = 7. How can we be playing the exact same two notes yet classifying them as different intervals? Thus we will revise our definition used for intervals once again.

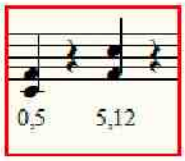
Definition 1.3: An interval class (IC) is defined as the shortest distance between two unordered pitch interval classes.

Then the 12 different intervals are now compressed into 6 different interval classes as shown in Table 1.2.

Table 1.2

Note played with 0	Pitch class intervals	Shortest distance	IC
1 and 11	PIC(0,1) = 1 (mod 12), PIC(1,0) = 11(mod 12)	1	1
2 and 10	PIC(0,2) = 2 (mod 12), PIC(2, 0) = 10(mod 12)	2	2
3 and 9	PIC(0,3) = 3 (mod 12), PIC(3,0) = 9(mod12)	3	3
4 and 8	PIC(0,4) = 4 (mod 12), PIC(4,0) = 8 (mod 12)	4	4
5 and 7	PIC(0,5) = 5 (mod 12), PIC(5,0) = 7 (mod 12)	5	5
6	PIC(0,6) = 6 (mod 12), PIC(6,0) = 6 (mod 12)	6	6

Then we can use the definition of interval classes to classify each pair of notes played together. The two intervals 5 and 7 in Example 1.2 are considered equivalent because they are both in interval class 5.



Section 1.2 Enumerating Music

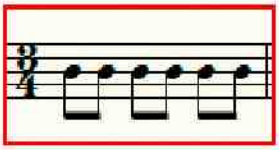
Depending on the limitations we put on a particular composition there are many things we can count using techniques from combinatorics all based on the overall question of how many different musical compositions are possible. Let's start with a few simple examples given by Hook [6].

Example 1.3: How many 6 note melodies are there using the 8 tones of a major scale? For each note we have 8 choices and we need to choose a total of 6 notes. Therefore there are 8^6 different melodies. How many 6 note melodies are there using the 8 tones of a major scale without repetition? Now for the first note we have 8 choices, the second note we have 7 choices, and so on. Then there are $8 \cdot 7 \cdot 6 \cdot 5 \cdot 4 \cdot 3$ melodies.

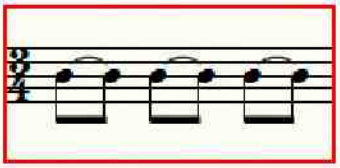
Example 1.4: How many melodies using the 8 tones of a major scale have at most a length of 6 notes? Now we need to count the number of 1 note melodies, 8, the number of 2 note melodies, 8^2, the number of 3 note melodies, 8^3, and so on. Then accounting for all the melodies from a length of 1 note to a length of 6 notes we have $8 + 8^2 + 8^3 + 8^4 + 8^5 + 8^6$. However, Hook points out that in order to be complete we should count the melody of length 0, so we have a total of $8^0 + 8 + 8^2 + 8^3 + 8^4 + 8^5 + 8^6 = 1 + 8 + 8^2 + 8^3 + 8^4 + 8^5 + 8^6$ melodies.

Hook also mentions that in general the formula for counting the number of melodies using n notes with a length of at most k notes is $\sum_{j=0}^{k} n^j$. Notice that this general formula counts "how many j note melodies are there using n notes?" for all values of j, from $0 \leq j \leq k$. Example 1.3 only counts the possible melodies for a specific length, $j = 6$, whereas Example 1.4 illustrates the use of this general formula in order to find all possible melodies for the values of j from $0 \leq j \leq 6$.

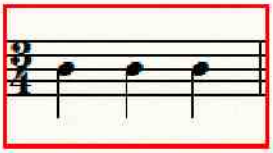
Example 1.5: How many different rhythms are possible in 3/4 measure, where there are only eighth notes or whole multiples thereof? Note that 3/4 measure means that there are three total beats in a measure and a quarter note is worth one beat. Since we are using eighth notes as the unit of measure for the possible rhythms we first notice there are six eighth notes spanning a 3/4 measure. Two eighth notes equal a quarter note, since three quarter notes span a 3/4 measure, we need six eighth notes.



To answer the question we need to consider the number of ways we can create different rhythms with these eighth notes and we can do this by connecting the eighth notes with a series of ties. In music, ties connect one note to a second note so that the second note is not played but just a sustaining of the first note. For example, we can add in three ties to our six eighth notes as shown.



These three ties make it so that the 2nd, 4th, and 6th eighth note are not played and the result is a rhythm that is equivalent to playing three quarter notes as shown below.



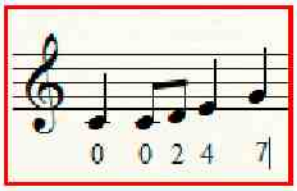
Then in order to count the number of possible rhythms using multiples of eighth notes we need to choose which eighth notes to connect with a tie. Accordingly we must choose connect or don't connect note 1 to note 2. Choose connect or don't connect note 2 to note 3 and so on. This leaves us with a choice of two, connect or don't connect, a total of 5 times. Then there are 2^5 different rhythms.

Section 1.3 Transformations and Twelve-Tone Technique

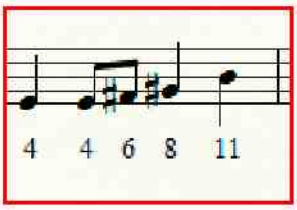
Given a specified melody there are different permutations of the melody that we can count. These permutations are based on the four possible transformations on a melody: transposition, retrograde, inversion, and retrograde-inversion. Let's take a moment to examine these transformations.

Definition 1.4: The transposition of a melody is created by adding a set number of half steps to each note in the melody. Thus transposition preserves all the distances between the notes and just changes the tone of the starting note and all the notes thereafter.

Example 1.7: Given the melody shown below we can perform a transposition to the melody by adding 4 half-steps to each note.

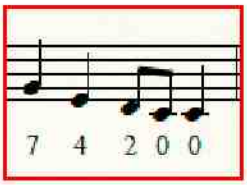


Our transposition then will be 0 + 4, 0 + 4, 2 + 4, 4 + 4, 7 + 4, which gives us the following melody as the result.



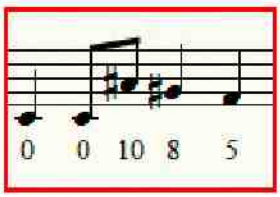
Definition 1.5: The retrograde of a melody is created by playing the melody backwards. This can be done by reversing the order of the notes played or reading the music from right to left instead of left to right.

Example 1.8: Given the same starting melody as in Example 1.7, the retrograde transformation reverses the order of the notes played to be 7, 4, 2, 0, 0 and looks like this:



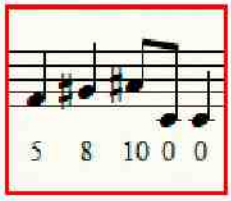
Definition 1.6: The inversion of a melody is created by replacing each tone with the result of subtracting each tone from 12. Then playing two notes with a pitch class interval of 3 gives us a pitch class interval of 9 (12 – 3) in its inversion. The result is playing the melody "upside-down."

Example 1.9: Given the same starting melody as in Example 1.7, we subtract each note from 12 $12 - 0 = 12 = 0 \pmod{12}$, $12 - 0 = 12 = 0 \pmod{12}$, $12 - 2 = 10$, $12 - 4 = 8$, $12 - 7 = 5$. Here is the result:



Definition 1.7: The retrograde inversion of a melody is created by subtracting each tone from 12 and then playing the sequence backwards. Thus the retrograde inversion is playing a melody upside-down and backwards.

Example 1.10: Given the same starting melody as in Example 1.7 we can take the inversion from Example 1.9 and play it backwards.



These four transformations, transposition, retrograde, inversion, and retrograde inversion, are used in a musical composition method called Twelve-tone technique, which is part of the inspiration for the algorithms created in Chapter 4. Twelve-tone technique is based on first establishing what is called a 12-tone row (sometimes just referred to as tone row).

Definition 1.8: A 12-tone row is a sequence of 12 notes where each of the 12 musical tones is played exactly once. If we wanted to count the total number of possible 12-tone rows, similar to Example 1.3, we count this as 12 possible.

The main purpose of twelve-tone technique is to have a method for composing a piece of atonal music (not written in any major or minor key) and this technique was created by Arnold Schoenberg in the early 1920s [9]. It consists of first starting with a 12-tone row sometimes called the prime series. Then to compose the rest of the piece certain rules must be followed [9]:

1. In every 12 note set, every tone must appear exactly once.
2. Every 12 note set can be the original 12-tone row (prime series) or any of the following transformations: inversion, retrograde, and retrograde inversion.

3. Every 12 note set does not have to start on the very first note in the set; the set can start on any of the notes.

With Twelve-Tone Technique, there are many possible things to count, including "how many permutations of the prime series are possible based on the three rules above?" and "how many unique tone rows are there after accounting for transformations?" The answers to these questions are explored in [12]. We briefly discuss Twelve-Tone Technique here to ensure that the reader is familiar with musical transformations and to introduce the idea of creating certain rules when composing music. These ideas will be employed in Chapter 4.

Chapter 2: Rhythmic Canons

In music, a canon (or round) is a musical piece produced by two or more voices playing the same melody, separated by a specific time interval. A rhythmic canon is specifically a canon where voices play a rhythm (instead of a melody) separated by a specific time interval.

There has been a wide range of mathematical research conducted on canons, both tonal and rhythmic. Rhythmic canons are perhaps a more simple type to research first, since only the rhythms are the focus and we can disregard the aesthetics behind different tone intervals being played together. This research focuses on counting different types of rhythmic canons using the counting techniques from Burnside's Lemma and Polya's Enumeration Theorem. The majority of this chapter is a summary of Hall and Klingsberg's research from [4] and [5].

2.1 Rhythm Patterns

We begin by discussing the major components that make a rhythmic canon, starting with a rhythm pattern. A rhythm pattern is defined as a sequence of note onsets over a set number of beats (which can be referred to as a measure).

We will denote a note onset with a "1" and a rest with a "0".

Example 2.1: Rhythm pattern: 10001010. This is a rhythm pattern that consists of eight beats; three beats contain note onsets and five beats are rests.

The rhythm patterns we will consider will be one periodic, meaning there is one sequence of note onsets that is then repeated continuously. Two rhythm patterns are equivalent if one pattern is a shift of the other. Rhythm patterns that are equivalent make up the equivalence classes, also called rhythm cycles.

Example 2.2: 10001010 and 10101000 are equivalent since the second pattern is the same as the first with a shift of four beats.

Since Example 2.2 consists of eight beats, there are eight rhythm patterns that make up the rhythm cycle, i.e. the following rhythm patterns are all equivalent:

10001010; 01000101; 10100010; 01010001; 10101000; 01010100; 00101010; 00010101

There are lots of variations that can be made when creating a rhythm pattern. The number of beats (eight), the number of note onsets (three), as well as the spacing between the note onsets can all be adjusted. Based on these three specifications (number of beats, number of note onsets, and possible spacing between note onsets), we can count the number of possible rhythm patterns, and more importantly, we can count the number of possible distinct rhythm patterns (the rhythm cycles).

2.2 Counting Rhythm Cycles

Approach 1: With the specification of n total beats and k note onsets (meaning there are $n \quad k$ places of "rest"), our first approach will be to count how many ways we can place the k note onsets over n beats. We can count this as n choose k. Remember that two rhythm patterns are equivalent if one is a shift of the other. Then since we can shift each pattern a total of n times, we need to divide by n in order to avoid overcounting. Then we get the total number of rhythm cycles with n total beats and k note onsets to be $\binom{n}{k} \Big/ n$.

Example 2.3: Count the total number of rhythm cycles with 8 total beats and 3 note onsets.

Using $\binom{n}{k} \Big/ n$, we get $\binom{8}{3} \Big/ 8 = 56/8 = 7$. One can check by hand that the 7 distinct rhythm patterns are specifically: 11100000, 11010000, 11001000, 11000100, 11000010, 10101000, 10010100.

Example 2.4: Count the total number of rhythm cycles with 4 total beats and 2 note onsets.

Using $\binom{n}{k}\Big/n$, we get $\binom{4}{2}\Big/4 = 6/4$ which is not an integer value. Why did this occur? These values are small enough we can find the total number of rhythm cycles by brute force. We can list all six possibilities of 4 total beats with 2 note onsets: 1100, 1010, 1001, 0110, 0101, 0011. With our reasoning from above, we can take 1100 and shift it 4 times so that we have an equivalence class for 1100 consisting of 4 elements: 1100, 0110, 0011, 1001. However, when we take 1010 and shift it 4 times we only have an equivalence class with 2 elements: 1010, 0101. This demonstrates the error in our reasoning. When we shift the rhythm patterns, the equivalence classes are not all the same size! Note that even though our formula gave us 6/4, after checking through brute force we have the total number of rhythm patterns being 2. Thus we need to revise our approach for counting these rhythm patterns.

Approach 2: For our second approach, we need to adjust the way we are counting the elements in the equivalence classes. We can think of the equivalence classes as orbits induced by a group action and apply Burnside's Lemma.

Burnside's Lemma *Let a finite group G act on a finite set S; for each $\beta \in G$, define $fix(\beta)$ to be the elements $s \in S$ such that $\beta \cdot s = s$. Then the number of orbits that G induces on S is given by*

$$\frac{1}{|G|}\sum_{\beta \in G} |\, fix(\beta)\,|.$$

Let $G = \{r^i \mid i = 0...n-1\}$, where r^i denotes shifting the rhythm pattern i places. Let S be the set of all possible rhythm patterns for n total beats with k note onsets. Then for each $r^j \in G$, we will define $fix(r^j)$ to be the elements $s \in S$ such that $r^j \cdot s = s$. Then the number of equivalence classes that G induces on S is given by $\frac{1}{|G|}\sum_{r^j \in G} |\, fix(r^j)\,|$.

Then we can apply this new approach to Example 2.4. For n 4, we have $G = \{r^0, r^1, r^2, r^3\}$. Notice that r^0 is the identity so for n–4 and k–2, we have r^0–{*1100, 1010, 1001, 0110, 0101, 0011*}. Then $|\ fix(r^0)| = 6$. When we examine the elements of r^1, we are looking for elements that remain the same after a shift of 1. Since no rhythm patterns are the same after a shift of 1, r^1 is empty. Also, note that r^3 is shifting the rhythm pattern 3 times to the right, which is equivalent to shifting the rhythm pattern 1 time to the left, so r^3 is also empty. Then $|\ fix(r^1)\ |$ and $|\ fix(r^3)| = 0$. There are two elements in r^0 that are the same after a shift of two, so r^2–{*1010,0101*}. Then $|\ fix(r^2)| = 2$. Then we can count the total number of rhythm patterns using Burnside's Lemma:

$$\frac{1}{4}(|\ fix(r^0)\ | + |\ fix(r^1)\ | + |\ fix(r^2)\ | + |\ fix(r^3)\ |) = \frac{1}{4}(6+0+2+0) = \frac{1}{4}\cdot 8 = 2.$$

Notice we can now apply this second approach to Example 2.3 to better understand why this new approach works. Let $G = \{r^0, r^1, r^2, r^3, r^4, r^5, r^6, r^7\}$. We can calculate $|\ fix(r^0)| = 56$ since there are *8 choose 3* ways to place 3 note onsets over 8 beats. When we shift a rhythm pattern 1 place, in order for it to remain the same each term needs to match the term in front of it which would be the same as saying each term needs to match the term behind it. Note that in general when we consider shifting a pattern q places we can shift q places to the left or right, so a shift of q places is equivalent to a shift of n q places. Then shifting 7 places is the same as shifting 1 place and there are no rhythm patterns that stay the same in this case so $|\ fix(r^1)| = 0 = |\ fix(r^7)\ |$. When we shift a rhythm pattern 2 places, in order for it to remain the same each term needs to match two places ahead, i.e. if there is a 1 on the 1st beat there must be a 1 on the 3rd beat, 5th beat, and 7th beat so that when it is shifted 2 places it remains the same. This is impossible with only 3 note onsets so $|\ fix(r^2)| = 0 = |\ fix(r^6)\ |$. Similarly it is impossible

with only 3 note onsets for the rhythm pattern to remain the same after a shift of 3 places and a shift of 4 places so $|\mathit{fix}(r^3)| = 0 = |\mathit{fix}(r^5)|$ and $|\mathit{fix}(r^4)|$.

Thus we can count the total number of rhythm patterns using Burnside's Lemma:

$$\frac{1}{8}(|\mathit{fix}(r^0)|+|\mathit{fix}(r^1)|+|\mathit{fix}(r^2)|+|\mathit{fix}(r^3)|+|\mathit{fix}(r^4)|+|\mathit{fix}(r^5)|+|\mathit{fix}(r^6)|+|\mathit{fix}(r^7)|) = \frac{1}{8}(56) = 7.$$

2.3 Asymmetrical Rhythms

An interesting type of rhythm pattern to investigate is the asymmetrical rhythm. For mathematical purposes, rhythm cycles are defined as asymmetric if they cannot be broken into ℓ equal parts where each part starts with a note onset. In other words, asymmetric rhythm cycles are rhythms that contain rests on at least one of the "downbeats," where the downbeat is the first beat of each ℓ equal part.

Example 2.5: Let there be a rhythm cycle of length 4 and let $\ell = 2$. Then out of the six different rhythm patterns (see Example 2.4) we have three that are asymmetrical rhythms: 1000; 1100; 0000. Notice that we cannot use 3 or 4 note onsets because we would be able to break the 4 beats into 2 equal parts, and each part would start with a note onset.

At first glance, asymmetrical rhythms may be associated with the musical term "syncopation" or playing on the "upbeat" as opposed to the downbeat, but note that the mathematical definition of asymmetrical allows for note onsets to be played on a "downbeat" but specifies that at least one downbeat has a rest. Example 2.6 shows that simply avoiding downbeats and playing only on upbeats is not enough to be asymmetrical.

Example 2.6: With 8 total beats and $\ell = 4$, the rhythm pattern 10101010 contains four note onsets, all on the downbeats of the rhythm cycle, where as 01010101 contains four note onsets

on what are called the upbeats. Observe that based on our definition of rhythm cycles these two patterns are equivalent by a shift of 1 and the rhythm pattern is not asymmetrical.

Hall and Klingsberg's research on asymmetrical rhythms produced methods for counting the number of asymmetric rhythm cycles given a specified cycle length (i.e. period), starting with a period of $2n$. Before talking about Hall and Klingsberg's theorems, we need to introduce the use of function notation for these rhythms.

2.4 Rhythms as Functions

A rhythm pattern can be represented as a function $f: Z \rightarrow \{0, 1\}$, where $f(x) = 1$ if there is a note onset on pulse x and $f(x) = 0$ otherwise. If a function is periodic with a period p, then $f(x) = f(x + p)$ for all x *in* Z. This means f has the domain $\boldsymbol{Z/pZ}$ or $\boldsymbol{Z_p}$. We will refer to functions (rhythm patterns) as being equivalent based on applying a shift. Then we can say $f(x)$ is the same as $f(x+p)$ mod the shift.

Recall the rhythm pattern from Example 2.1, 10001010. Applying these definitions, for $x \in \{1,5,7\}$ we have $f(x) = 1$ since there is a note onset when $x \in \{1,5,7\}$ and $f(x) = 0$ for the remaining beats. Example 2.1 is periodic with a period of length 8, so $f(x) = f(x + 8)$ and has a domain $\boldsymbol{Z_8}$.

We will define a rhythm cycle to be an equivalence class of p-periodic functions mod the shift. For now, we will work with asymmetric rhythm patterns that can be divided into two parts. Then the period of the rhythm patterns must be even so we will let the period $p = 2n$. In order to be asymmetric, if a note onset occurs at beat x, no note onset occurs at beat $x + n$. In other words, if $f(x) = 1$ then $f(x + n) = 0$.

The rhythm pattern has a total of $2n$ elements (one element per beat) and we can partition all $2n$ elements into n pairs: $\{\{0, n\}, \{1, n+1\}\ldots\{n-1, 2n-1\}\}$. In order to be asymmetric, we have three possibilities for each of these pairs:

Pair Possiblities:

Option 1. Both $f(x)$ and $f(x-n)$ are both 0,

Option 2. The first element, $f(x)$, is 1 and the second element, $f(x+n)$, is 0,

Option 3. The first element, $f(x)$, is 0 and the second element, $f(x+n)$, is 1.

This leaves us with 3^n possible asymmetric rhythm patterns for a period $p = 2n$. However some of these 3^n patterns may be duplicates mod shift, so what we really care about is the total number of cycles (equivalences classes) which we can count using Burnside's Lemma.

Let us denote all 3^n possible rhythm patterns with the set S_{2n} where $S_{2n} = \{f : Z_{2n} \to \{0,1\} \mid f(x) = 1 \Rightarrow f(x+n) = 0\}$. We need to count the number of equivalence classes modulo the shift for S_{2n}. The equivalence classes of S_{2n} can be seen as orbits induced by a group action. The group is $G = Z_{2n}$ and an element r^m in G acts on a rhythm pattern by shifting it forward m beats. Namely for $f \in S_{2n}$, the orbit of f induced by r^m is the set $\{r^m \cdot f \mid r^m \in G\}$, where $(r^m \cdot f)(x) = f(x-m)$.

Example 2.7: Consider the rhythm cycle 10110000. The orbit of f induced by r^2 is {10110000, 00101100, 00001011, 11000010}.

Since the equivalence classes of S_{2n} can be seen as orbits we can apply Burnside's Lemma in Section 2.5 to help us count the number of asymmetrical rhythms. In Section 2.6, we will specify the number of note onsets we want in a rhythm cycle (r notes) and similarly we can use Burnside's Lemma to count the number of r-note asymmetric rhythms of period $2n$ by viewing the equivalence classes as orbits.

2.5 Hall and Klingsberg's Theorem 1

The following theorem provides us with a formula for finding the number of asymmetric rhythm cycles of period $2n$. The proof given by Hall and Klingsberg is summarized here and revolves around finding *fix(B)* (from Burnside's Lemma).

Theorem 1 *The number of asymmetric rhythm cycles of period 2n is given by*

$$\frac{1}{2n}\left[\sum_{d|n}\phi(2d)+\sum_{\substack{d|n\\ d\,odd}}3^{n/d}\phi(d)\right]$$

Where $\phi(d)$ *is the number of integers* $1 \le x \le d$ *such that x is relatively prime to d.*

Proof: Let the period be $2n$. Then for each divisor d of $2n$, we will first find the number of elements of order d. Let k have order d which means $kd = 2n$ where d is the smallest multiple of k that equals $2n$. Then $k=2n/d$ and the elements of order d in $\boldsymbol{Z_{2n}}$ are the subgroup of multiples of k *mod* $2n$. Define the multiples in this subgroup to be of the form $B=kj$, where $1 \le j \le d$ and $gcd(j,d) = 1$. Notice that if we multiply B by d we get $Bd = kjd$. Since k is of order d we have a multiple of k *mod* $2n$. We need $gcd(j,d) = 1$ because if j were a divisor of d, then k would have a smaller order than d. In fact the order would then be $d/gcd(j,d)$. Since we need $gcd(j,d) = 1$, there are $\phi(d)$ elements in this subgroup, where $\phi(d)$ is the number of integers in $\{2, \ldots, d\text{-}1\}$ that are relatively prime to d.

Now we will examine *fix(B)* and there are two cases:

Case 1: k divides n

Since k divides n, this leaves us with $f(x) = f(x + k)$. This implies that $f(x) = f(x + n)$ since k divides n. But in order to be asymmetric, $f(x)$ can equal $f(x+n)$ only if $f(x)=0$. Therefore for Case 1, we have $|fix(B)|=1$.

Case 2: k does not divide n

Even though k does not divide n, we do know that k divides $2n$ so the rhythm cycle is k-periodic. This means the asymmetry of the rhythm cycle will be determined by the values given to $\{0, \ldots, k\text{-}1\}$. We know $2n \equiv 0 \pmod{k}$ with $n \neq 0 \pmod{k}$. Then $2n \equiv k \pmod{k}$ and since k is even, we get $n \equiv \frac{k}{2} \pmod{k}$. Thus when we examine the n pairs in the rhythm pattern we really only need to examine $k/2$ pairs. We have three choices for each of these pairs, so this leaves us with $|fix(B)| = 3^{k/2}$. Since we know $dk = 2n$ for some d, we know $k/2 = n/d$. Therefore by substitution we have $|fix(B)| = 3^{n/d}$. □

To better illustrate this proof let's apply the proof to an example to explore the details of how it works:

Example 2.8: Let $n = 4$ so the period is 8. Then the possible divisors of 8, which we will sequentially denote as d, are 1, 2, 4, 8. Say $k = 2n/d$. We will find $\phi(d)$ for each possible value of d.

Now for $d = 1$, we need the multiples of k that have order 1 where $k = 8/1 = 8$. Then to find the number of elements of order 1, we need to find how many elements we have of the form $B = 8j$, where $1 \leq j \leq 1$ and $\gcd(j,1) = 1$. There is only one element, when $j = 1$, $B = 8$. Since we are counting how many elements of the form $B = 8j$, when $\gcd(j,1) = 1$ there will be $\phi(d)$ of them. Observe $\phi(1) = 1$.

Next for $d = 2$, we need the multiples of k that have order 2 where $k = 8/2 = 4$. Then to find the number of elements of order 2, we need to find how many elements we have of the form $B = 4j$, where $1 \leq j \leq 2$ and $\gcd(j,2) = 1$. There is only one element, when $j = 1$, $B = 4$. Note that $\phi(2) = 1$.

Next for $d = 4$, we need the multiples of k that have order 4 where $k = 8/4 = 2$. Then to find the number of elements of order 4, we need to find how many elements we have of the form $B = 2j$, where $1 \leq j \leq 4$ and $\gcd(j,4) = 1$. There are two elements, when $j = 1$, $B = 2$ and when $j = 3$, $B = 6$. Observe $\phi(4) = 2$.

Finally for $d = 8$, we need the multiples of k that have order 8 where $k = 8/8 = 1$. Then to find the number of elements of order 8, we need to find how many elements we have of the form $B = 1j$, where $1 \leq j \leq 8$ and $\gcd(j,8) = 1$. There are four elements, when $j = 1$, $B = 1$, when $j = 3$, $B = 3$, when $j = 5$, $B = 5$ and when $j = 7$, $B = 7$. Note that $\phi(8) = 4$.

Let's summarize our findings so far:

Table 2.1	k	B	# of elements of order $d - \phi(d)$
$d = 1$	8	8	1
$d = 2$	4	4	1
$d = 4$	2	2,6	2
$d = 8$	1	1,3,5,7	4

Now we need to examine $fix(B)$ for all value of B in Table 2.1. We will examine these based on two cases:

Case 1: k divides n

Since $n = 4$, we have $d = 2$, 4, and 8 to examine. Since k divides n, this leaves us with $f(x) - f(x + k)$. Since we know k is a multiple of n this means that $f(x) = f(x - n)$. For instance, when $d = 2$ we have $k = 4$. Thus $f(x) - f(x \cdot 4)$ which gives us $f(x) - f(x \cdot 4)$ automatically. When $d = 4$ we have $k = 2$ and we know $f(x) - f(x + 2)$. Since 4 is a multiple of 2, this implies that $f(x) - f(x - 4)$. When $d = 8$ we have $k = 1$ and we know $f(x) = f(x + 4)$. Since 4 is a multiple of 1, this

implies that $f(x)$ $f(x \cdot 4)$. Then for all values of k that divide n we have $f(x)$ $f(x \cdot 4)$. For asymmetrical rhythms this only holds if $f(x) = 0$. There is only 1 possible rhythm pattern with $f(x) = 0$ for all x, so $fix(B) = 1$ for all B when k divides n.

Case 2: k does not divide n

For Case 2, we are left with $d = 1$ to examine. Since $k = 8$ does not divide $n = 4$, we will have to examine the 8/2 pairs and make sure that they create an asymmetrical rhythm pattern based on Options 1, 2, and 3 for pair possibilities. Then $|fix(B)|$ $3^{8/2}$. Since we know dk $2n$ for $d = 1$, we know $k/2 = n/d$. Therefore by substitution we have $|fix(B)| - 3^{4/1}$.

Now we have the following formula for counting asymmetrical rhythms as our result

$$\frac{1}{2n}\left[\sum_{d\,n}\phi(2d)+\sum_{\substack{d\,n\\ d\,odd}}3^{n/d}\phi(d)\right]. \square$$

Using this formula to count the number of asymmetrical rhythms for $n = 4$ we have

$$\frac{1}{8}\left[\sum_{d\ 1,2,4}\phi(2d)+\sum_{d\ 1}3^{n/d}\phi(d)\right]=\frac{1}{8}\left[\phi(2)+\phi(4)+\phi(8)+3^{4/1}\phi(1)\right]=\frac{1}{8}\left[1+2+4+3^{4/1}\cdot 1\right]=11$$

Here are the eleven asymmetrical rhythms that are possible when $n = 4$:

10000000, 11000000, 10100000, 10010000, 11100000, 11110000, 00000000, 11010000, 10110000, 1001010, 10110100.

2.6 Hall and Klingsberg's Theorem 2

The following theorem gives a formula for finding the number of asymmetric r-beat rhythm cycles, where r is a specified number of note onsets. It is important to notice if $r = 0$ there is one obvious cycle, so this theorem sets $r \geq 1$. Hall and Klingsberg's proof, summarized below, is similar to the proof of Theorem 1 and revolves around finding $fix(B)$.

Theorem 2 *For any* $1 \le r \le n$, *the number of asymmetric r-beat rhythm cycles is given by*

$$\frac{1}{2n}\sum_{\substack{d \mid \gcd(n,r) \\ d \text{ odd}}} \phi(d)\binom{n/d}{r/d}2^{r/d}.$$

First for each divisor d of $2n$, we need to find the number of elements of order d (there are $\phi(d)$ of them). Next we determine $|fix(B)|$ and there are 3 cases to consider:

Case 1: k divides n

Similar to Case 1 from the proof of Theorem 1, this implies $f(x)$ $f(x \cdot n)$. Hence $f(x)$ must equal 0 in order to by asymmetric. If $f(x) = 0$ for each x, then this means $r = 0$, so for Case 1, $|fix(B)| = 0$.

Case 2: k doesn't divide n and d doesn't divide r

In order to be k-periodic, the number of elements x such that $f(x) = 1$ would have to be a multiple of d. But r isn't a multiple of d, so for Case 2, $|fix(B)| = 0$.

Case 3: k doesn't divide n and d does divide r

Now we can look at the n/d pairs $\{0, k/2\}\ldots\{k/2 - 1, k - 1\}$ and make sure there are exactly r 1s. In order to make this an asymmetric rhythm, we know there are three choices for each of those pairs and in order for there to be r 1s we must have exactly r/d pairs with note onsets. Now we must use the three choices given in the "Pair Possibilities" outlined in Section 2.4 and ensure that r/d pairs are given a note onset. To ensure this we will first choose r/d pairs out of n/d pairs. There are (n/d choose r/d) ways to do this. Next we will assign each of these chosen r/d pairs with Option 2 or Option 3 and there are $2^{r/d}$ ways to do this. Finally all of the remaining pairs will be assigned Option 1. Therefore for Case 3, $|fix(B)| =$ (n/d choose r/d)$*2^{r/d}$. □

2.7 Hall and Klingsberg: Further Results

Hall and Klingsberg were able to apply their findings on asymmetrical rhythms of period $2n$ to several different areas, including the complements of rhythm cycles, asymmetrical rhythms of period ℓn and tiling canons.

The complement of a rhythm cycle simply implies swapping all the beats with rests and vice versa. The maximum number of beats in an asymmetric rhythm cycle is n and asymmetric rhythm cycles with n notes are complementary (they are equivalent to their own complements – by shifting). If we take Hall and Klingsberg's Theorem 2 and plug in $r - n$, we get a corollary that tells us how many complementary asymmetric rhythm cycles there are for cycles with n notes.

Hall and Klingsberg's Corollary: The number of r-note, l-asymmetric rhythms is

$$\frac{1}{\ell n}\sum_{\substack{d \gcd(n,r) \\ \gcd(d,\ell)=1}} \phi(d)\binom{n/d}{r/d}\ell^{r/d}. \qquad (*)$$

Throughout this chapter, we only considered asymmetrical rhythms of period $2n$. We can now use a more general definition for asymmetrical rhythms. Let a periodic rhythm of period ℓn be ℓ-asymmetric if $f(x) = 1$ implies $f(x + n) = 0$. Hall and Klingsberg's Theorems 1 and 2 can be modified so that they count the number of asymmetric rhythm cycles with l-asymmetry instead of 2-asymmetry.

Hall and Klingsberg's research is taken a step further to enumerate specific kinds of tiling canons. A canon (or round) is a musical figure produced when two or more voices play the same melody, with each voice offset by a specific time interval. A rhythmic tiling canon is a rhythmic canon with the restriction that when all voices are played, the resultant rhythm has exactly one

note onset per unit beat. Hall and Klingsberg examine the cycles of 12-periodic rhythmic tiling canons, with $\ell = 3$, $n = 4$, and $r - n = 4$, where ℓ is the number of parts the period is divided into, n is the number of beats in each ℓ part, and r is the number of note onsets. They show that there are exactly eight cycles with 4-note onsets ($r - 4$). In general, any ℓn-periodic rhythm with n note onsets that is ℓ-asymmetric forms a tiling canon of ℓ voices, offset by multiples of n notes. The number of these tiling canons can be found by substituting $r - n$ in (*) (Theorem 2 modified). It is important to recognize that their results provide a way to enumerate tiling canons that have periodic inner and outer rhythms and equally-spaced onsets for the outer rhythm.

Chapter 3: Crab Canons

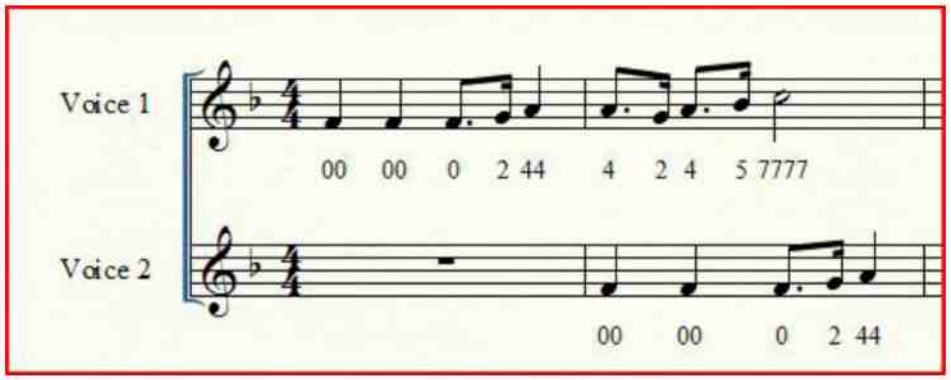
Have you ever sung "Row, Row, Row Your Boat" in a round with multiple people? Then you have been a part of what musicians call a melodic canon. Remember that a canon (or round) is a musical piece produced by two or more voices playing the same melody, separated by a specific time interval. Previously in Chapter 2 we discussed research on rhythmic canons (specifically asymmetrical rhythms). In this chapter, we will look more closely at melodic canons including a discussion on the different types of canons and an analysis of Bach's well known crab canon from Bach's *Musical Offering*.

3.1 Types of Canons

There are many types of melodic canons composed and performed by musicians. Perhaps the simplest type of melodic canon is playing strictly unison, where the second player is repeating the first player only separated by a specified time interval.

Example 3.1 Unison:

Here are the first two measures of "Row Your Boat" that we will use for our example of a unison canon:



This would be considered a unison canon since the two voices have the same melody. The second voice starts at the beginning of the melody after the first voice finishes the first measure. Notice that the time signature is 4/4 (four beats a measure with a quarter note equaling one beat), so the second voice is four beats behind the first voice.

In order to analyze the intervals being played, we let every 8^{th} note (rounded because of the dotted eighth notes) get one value, and we enumerate the music based on Table 3.1. There are 12 possible notes and since we are in the key of F, we let F = 0 and use the values 0 through 11 to enumerate all 12 possible notes:

Table 3.1

Note	Value
F	0
F#	1
G	2
G#	3
A	4
A#	5
B	6
C	7
C#	8
D	9
D#	10
E	11

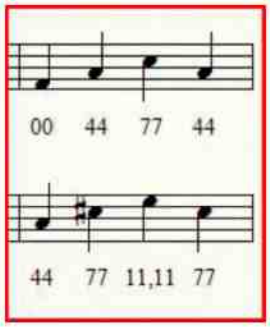
Once we assign a value 0-11 to each note we can find the intervals (in half steps) being played by the two voices by simply subtracting the values assigned to each voice and then label each with an interval class (see Chapter 1).

Player 1	Player 2	interval in half steps	interval class
4	0	4	4
2	0	2	2
4	0	4	4
5	0	5	5
7	0	7	5
7	2	5	5
7	4	3	3
7	4	3	3

Besides unison, other types of melodic canons include interval canons, canons in contrary motion, and canons with any combination of musical permutations including retrograde, inversion, and retrograde inversion (see Chapter 1 Section 1.3) as well as augmentation (increasing the intervals played) and diminution (decreasing the intervals played).

Interval canons are similar to unison except the second player plays a different melody following the first player's melody. Also the two voices are separated by a specified note interval in addition to a specified time interval. Hence the name "interval canon" is based on the separation by a specified note interval.

Example 3.2 Interval Canon: Here is a short example of what an interval canon might look like with an interval separation of 4 half steps. Every half beat is assigned a number according to its note value. Thus every quarter note is assigned two numbers.

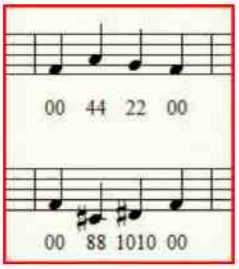


Notice that all four intervals are four half steps apart and the following table summarizes the types of intervals (in half-steps) we attain from the example (all interval class equal to 4).

Player 1	Player 2	interval in half steps	interval class
0	4	4	4
0	4	4	4
4	8	4	4
4	8	4	4
7	11	4	4
7	11	4	4
4	8	4	4
4	8	4	4

Canons in contrary motion (also called inverted canons) involve two players playing the same melodic intervals but moving in opposite directions. For example, the first and second players may start unison and then for the second note the first player may move 4 half steps up while the second player will move 4 half steps down.

Example 3.3 Contrary Motion: Here is a short example of a contrary motion canon:



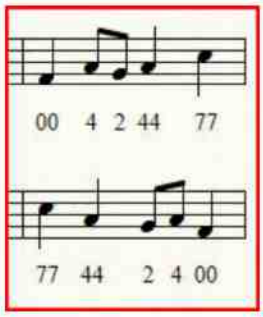
In this measure both voices start unison, but Player One moves up 4 half steps while Player Two moves down 4 half steps. For the third and fourth notes, Player One moves down 2 half steps and Player Two moves up 2 half steps. The following table summarizes the types of intervals (in half-steps) we obtain from the example.

Player 1	Player 2	interval in half steps	interval class
0	0	0	0
0	0	0	0
4	8	4	4
4	8	4	4

2	10	8	4
2	10	8	4
0	0	0	0
0	0	0	0

A crab canon (the focus of Sections 3.3 and 3.4) is a canon in which the melody is played both forwards and backwards. The crab canon is sometimes called a retrograde canon because retrograde refers to playing a melody backwards.

Example 3.4 Crab Canon: Here is a short example of a crab canon:



Notice that the second line is the same as the first line backwards. The following table summarizes the types of intervals (in half-steps) we attain from the example.

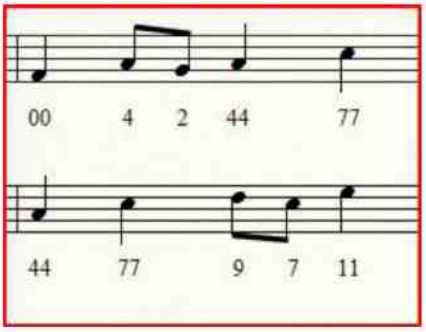
Player 1	Player 2	interval in half steps	interval class
0	7	7	5
0	7	7	5
4	4	0	0
2	4	2	2
4	2	2	2
4	4	0	0
7	0	7	5
7	0	7	5

Notice the symmetry in the interval classes that is created by both voices playing the same melody in opposite directions. As a result, the first half of the measure contains the interval class sequence 5, 5, 0, 2, and the second half of the measure contains the same interval class sequence but in reverse: 2, 0, 5, 5.

The last type of canon we will discuss is the mirror canon. Mirror Canons use both retrograde and inversion to create a canon that is played upside down and backwards by the

second player. The music for mirror canons can be placed on a table with the first and second player sitting across from each other which is why they are sometimes referred to as table canons.

Example 3.5 Mirror Canon: Here is a short example of a mirror canon where every half beat is assigned a number according to its note value:



Both players can read the music using the first line only. The second line for Player Two is shown for clarity. The first voice plays the first line as written. The second voice reads the music from the first line upside down and backwards which can easily be done when both voices sit across the table from each other. Looking at the first line the second voice will play backwards and inverted. Since we are using values 0 through 11 for the 12 different tones, we can subtract each value from 11 to find its inversion. Then the first note for the second voice is $|7 - 11| = 4$. The second note is $4 - 11 = 7$. Similarly we obtain 9, 7, and 11 for the final notes. The following table summarizes the types of intervals (in half-steps) we attain from the example.

Player 1	Player 2	interval in half steps	interval class
0	4	4	4
0	4	4	4
4	7	3	3
2	7	5	5
4	9	5	5
4	7	3	3
7	11	4	4
7	11	4	4

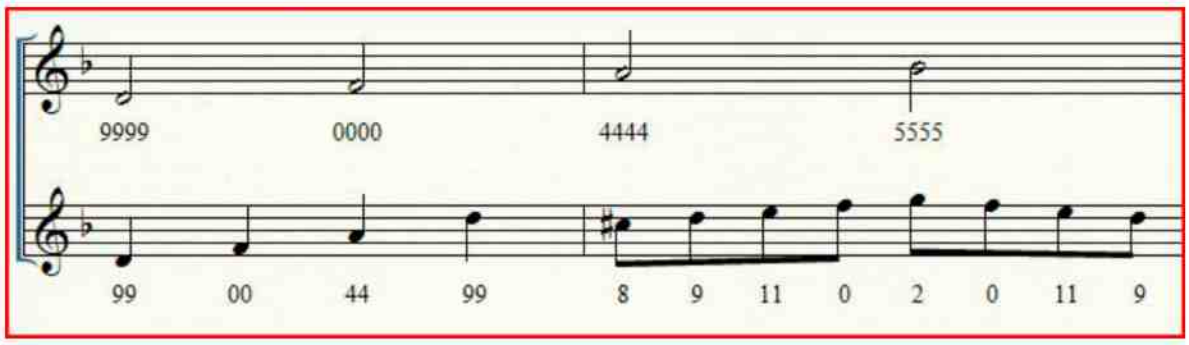
The reader should now have a general understanding of the different types of musical canons but certainly there are other types of canons that we have not mentioned. In addition, any combination of these types of canons can be used in a musical composition.

3.2 Well-Known Canons

Some other well-known canons (besides Row, Row, Row Your Boat) include Bach's Inventions, Pachelbel's Canon, and Bach's Musical Offering. Bach's Inventions is a well-known work of piano pieces consisting of 15 individual pieces. The theme of each Invention is a two-voice canon and each piece includes both the theme and a development of the theme which may no longer be a canon depending on the specific piece. Pachelbel's Canon, often played at weddings, consists of a three-voice canon. Bach's Musical Offering is a collection of musical pieces that includes ten canons, with "2 Canon *a 2*" being a notable example of a crab canon which we will look at in more depth in Section 3.3.

3.3 Crab Canons

The type of canon that is of interest for our research is called a retrograde or crab canon. A crab canon consists of playing a melody forward and backward at the same time. In other words the first player starts at the beginning of the piece and plays to the end and the second player starts at the end and plays the melody backwards (from right to left) until the player reaches the beginning. The well-known, Bach's "2 Canon *a 2*" from Musical Offering is an example of a crab canon. Notice that player 2 plays the same melody as player 1 but backwards. An analysis of the intervals played by the two voices was done by first numbering each tone with its assigned value 0 through 11 where F = 0 (see Table 3.1). The first two measures are analyzed here and for the full analysis see Appendix A.



Player 1	Player 2	interval in half steps	interval class
9	9	0	0
9	9	0	0
9	0	9	3
9	0	9	3
0	4	4	4
0	4	4	4
0	9	9	3
0	9	9	3
4	8	4	4
4	9	5	5
4	11	7	5
4	0	4	4
5	2	3	3
5	0	5	5
5	11	6	6
5	9	4	4

Notice that the interval class of 1 (see Chapter 1), which consists of an interval of 1 (minor 2nd) and 11 (major 7th) never occurs in the crab canon since this interval class consists of intervals that are only a half-step away and dissonant to each other.

3.4 Crab Canon Isomorphism

Crab canons consist of two voices playing simultaneously, with one voice playing the melody and the second voice playing the melody backwards (reading the music right to left). The two voices playing will create a sequence of intervals that the listener hears throughout the whole piece. Mathematically speaking we will say that any crab canons that have the same

sequence of intervals played are equivalent or isomorphic to each other. Disregarding the two voices having different tonality, player 1 playing note C and player 2 playing note E will be the equivalent to player 1 playing note E and player 2 playing note C. Here is my formal definition for crab canon isomorphisms which will be a key definition for the algorithms discussed in Chapter 4:

Definition 3.1: Let A and B be two crab canon sequences with n notes, where $A = \{a_1, a_2, \ldots, a_n\}$ and $B = \{b_1, b_2, \ldots, b_n\}$. Then $A \cong B$ if

a) $a_i = b_i$, or

b) $a_i = b_{n-(i-1)}$ and $a_{n-(i-1)} = b_i$.

Note that this definition does not take transpositions of a crab canon sequence into account, but that is not needed for ongoing analysis.

Example 3.1: Using the definition above, we will illustrate that the following two sequences with $n=6$ are isomorphic. Let $A = \{c, d, e, f, g, h\}$ and let $B = \{h, d, f, e, g, c\}$.

To check whether A and B are isomorphic we need to check the criteria from Definition 3.1 with all possible pairs of notes. Since $n = 6$, there are 6/2 = 3 pairs to check.

The first pair we will check is the 1st note and the 6 - (1 - 1) = 6th note. In A, c = 1st note and h = 6th note. This is equivalent to the 6th and 1st note in B, where h = 1st note and c = 6th note. The second pair we will check is the 2nd note and the 6 – (2 – 1) = 5th note. In A, d = 2nd note and g = 5th note. This is the same as the 2nd and 5th note in B, so they are equivalent because the notes are identical. The third and last pair we will check is the 3rd note and the 6 – (3 – 1) = 4th note. In A, e = 3rd note and f = 4th note. This is equivalent to B, where f = 3rd note and e = 4th note.

Therefore A and B are isomorphic.

Notice there will be $n/2$ pairs of notes that are played simultaneously in one crab canon sequence. Then to create an isomorphism for any given crab canon, we will have to choose whether to switch the notes for a_i and $a_{n-(i-1)}$. This gives us two choices, "switch" or "don't switch," and we will have to make this choice $n/2$ times. This means there will be $2^{n/2}$ possible isomorphisms for each crab canon. Therefore if we create an exhaustive list of all possible crab canons, we will need to divide by $2^{n/2}$ so that we do not overcount.

Example 3.2: Let A be a crab sequence with 6 notes where $A = \{1\ 2\ 3\ 4\ 5\ 6\}$. Notice that there are 3 pairs of notes that will be played simultaneously. Then there will be 2^3 possible isomorphisms and we can find them all by counting the number of ways to "switch" or "don't switch" the position of the note pairs that are played simultaneously.

Let 0 be "don't switch" and let 1 be "switch." Accordingly these are all equivalent to A.

000 $B = \{1\ 2\ 3\ 4\ 5\ 6\}$

100 $C = \{6\ 2\ 3\ 4\ 5\ 1\}$

010 $D = \{1\ 5\ 3\ 4\ 2\ 6\}$

001 $E = \{1\ 2\ 4\ 3\ 5\ 6\}$

110 $F = \{6\ 5\ 3\ 4\ 2\ 1\}$

011 $G = \{1\ 5\ 4\ 3\ 2\ 6\}$

101 $H = \{6\ 2\ 4\ 3\ 5\ 1\}$

111 $I = \{6\ 5\ 4\ 3\ 2\ 1\}$

In Chapter 4, we will discuss algorithms that will be used to create and count the number of possible ways to have a crab sequence under certain specifications. The algorithms will rely heavily on Definition 3.1 of isomorphisms in order to count only unique crab sequences.

Chapter 4: Crab Canon Algorithms

In this chapter, we will discuss two separate algorithms that can be used to create crab canons. These algorithms are limited by certain restrictions on the tones and rhythms with inspiration for these restrictions take from Twelve-tone Technique (see Chapter 1). Through the use of *Maple*, the algorithms will provide an exhaustive list of the possible 8-note sequences given the specified conditions. These 8-note sequences can then be used to create a unique crab canon piece.

When composing a crab canon, the goal is to create a melody that can be played forwards and backwards simultaneously without any dissonance. Let the rhythms played be all of the same length, a quarter note, and let the notes be the seven tones from any major key. For our examples, we will use the seven tones used in C major: C, D, E, F, G, A, B.

4.1 Creating the Algorithms

For Algorithm #1 we will let the canon have a length of 8 notes. Let the first and last notes both be C, and let the other six notes be chosen from the remaining six notes, D through B. We will enumerate these tones (as we numbered the notes used in Chapter 3) as follows:

Table 4.1

Note	Value
C	0
D	2
E	4
F	5
G	7
A	9
B	11

Now since we want to create a crab canon without any dissonance, we will be restricted to the interval classes 0, 3, 4, and 5, since in music, the interval classes 1, 2, and 6 are considered dissonant [2]. Remember these interval classes represent half step intervals, where 0 is the unison or octave, 3 represents three half steps or nine half steps, 4 represents a four half steps or eight half steps, and 5 represents five half steps or seven half steps (see Chapter 1, Table 1.2). Since we are restricting the possible intervals in our crab canon, the algorithm we need will incorporate these specified intervals. Thus if we examine the note D = 2, we can check to see which notes are allowed to be played with D by checking the interval class. D cannot be played with C or E since they are one half step apart. D (= 2) can be played with F (= 5) since they are in interval class 3 (5 – 2 = 3). Note D can also be played with G (interval class 5), A (interval class 5), and B (interval class 3). Using the numbers from Table 4.1 this is the same as saying note 2 can be played with notes 5, 7, 9, and 11. Then we can denote the possible note pairings with note D as: 2→{5,7,9,11}.

Based on these interval restrictions for all 7 tones, we have the following possible note pairings for our crab canon:

2→{5,7,9,11}

4→{7,9,11}

5→{2,9}

7→{2,4,11}

9→{2,4,5}

11→{2,4,7}

The enumeration above was used so that the reader can transition easily from the examples used in Chapters 1 and 3; however for the rest of this chapter (and for both algorithms),

it was simpler to use the values 0 through 6 to represent the 7 possible tones. Please note for the rest of this chapter we will be using the following enumeration:

Table 4.2	
Note	Value
C	0
D	1
E	2
F	3
G	4
A	5
B	6

Then the possible note pairings that will be checked in Algorithm #1 can be easily converted to:

1→{3,4,5,6}

2→{4,5,6}

3→{1,5}

4→{1,2,6}

5→{1,2,3}

6→{1,2,4}

Based on the restrictions we have set, the crab canon will be an 8 note sequence with the first and last note being C (= 0). Let the 8 note sequence be {0, *a, c, e, f, d, b*, 0} where 0 represents the note C, *a* represents the second note played, *c* represents the third note played, etc. Thus when two voices play this sequence together as a crab canon we have the following result:

Player 1	0	*a*	*c*	*e*	*f*	*d*	*b*	0
Player 2	0	*b*	*d*	*f*	*e*	*c*	*a*	0

Then the notes that will be played together based on this sequence are (0, 0), (*a, b*), (*c, d*), and (*e, f*) and we want the intervals from these notes to belong to interval classes 0, 3, 4, or 5.

Algorithm #1 will be used to find the possible sequences {*a, c, e, f, d, b*} based on these interval restrictions. Remember that the algorithm also needs to account for isomorphisms. As defined in Chapter 3, let A and B be two crab canon sequences with n notes, where $A = \{a_1, a_2, \ldots, a_n\}$ and $B = \{b_1, b_2, \ldots, b_n\}$. Then $A \cong B$ if

a) $a_i = b_i$, or

b) $a_i = b_{n-(i-1)}$ and $a_{n-(i-1)} = b_i$.

As discussed in Chapter 3, the sequence {*a, c, e, f, d, b*} is equivalent to all of these sequences: {*b, c, e, f, d, a*}, {*a, d, e, f, c, b*}, {*a, c, f, e, d, b*}, {*b, d, e, f, c, a*}, {*b, c, f, e, d, a*}, {*a, d, f, e, c, b*}, and {*b, d, f, e, c, a*}.

Here is an outline of Algorithm #1. The formal algorithm as written using Maple is in Appendix B.

4.2 Outline of Algorithm #1

We need to find all the possible ways to use 1 through 6 in positions "*a-f*" so that each number is only used once. Remember *a* represents the second note played, *b* represents the seventh note, *c* represents the third note, etc.

Step 1: First choose *a* from {1,2,3,4,5,6}.

Step 2: If *a* = 1, choose *b* from {3,4,5,6}.

If *a* = 2, choose *b* from {4,5,6}.

If *a* = 3, choose *b* from {1,5}.

If *a* = 4, choose *b* from {1,2,6}.

If *a* = 5, choose *b* from {1,2,3}.

If $a = 6$, choose b from $\{1,2,4\}$.

Step 3: Next choose c from $\{1,2,3,4,5,6\}$ but check to make sure that $c \neq a, c \neq b$.

Step 4: If $c = 1$, choose $d = 3$. If 3 has been used, choose $d = 4$. If 4 has been used, choose $d = 5$. If 5 has been used, choose $d = 6$.

If $c = 2$, choose $d = 4$. If 4 has been used, choose $d = 5$. If 5 has been used, choose $d = 6$.

If $c = 3$, choose $d = 1$. If 1 has been used, choose $d = 5$.

If $c = 4$, choose $d = 1$. If 1 has been used, choose $d = 2$. If 2 has been used, choose $d = 6$.

If $c = 5$, choose $d = 1$. If 1 has been used, choose $d = 2$. If 2 has been used, choose $d = 3$.

If $c = 6$, choose $d = 1$. If 1 has been used, choose $d = 2$. If 2 has been used, choose $d = 4$.

Step 5: Next choose e from 1 to 6, but e cannot equal a, b, c, or d.

Step 6: Finally let f be from 1 to 6, whatever value hasn't been used yet.

Step 7: Check that (e, f) is a possible note pairing.

Repeat steps until all possible choices are used. If during Steps 2, 4, or 7 the result is not a possible note pairing, then throw it out and return to Step 1.

4.3 Results of Algorithm #1

The algorithm gave a total of 144 different sequences. For the full list see Appendix B. We know that there are 4 pairs of notes that will be played simultaneously in our crab canon sequence; however the first pair is played in unison (since the first and last notes are both C). Then when counting the isomorphisms, there are really only 3 pairs of notes that we will need to account for. Remember from Chapter 3 we can count all of the isomorphisms by counting the number of ways to "switch" or "don't switch" the position of the note pairs that are played simultaneously (see Section 3.4). Therefore there are 2^3 possible isomorphisms for each crab canon sequence. Since the Maple output gave us 144 different sequences, there are precisely 144/8 = 18 unique crab canon sequences based on Algorithm #1.

The 18 unique crab canon sequences are listed from the Maple output in the form $\{a, b, c, d, e, f\}$ are {1, 3, 2, 5, 4, 6}, {1, 3, 4, 6, 2, 5}, {1, 4, 2, 6, 3, 5}, {1, 4, 3, 5, 2, 6}, {1, 6, 2, 4, 3, 5}, {1, 6, 3, 5, 2, 4}, {2, 4, 1, 6, 3, 5}, {2, 4, 3, 5, 1, 6}, {2, 5, 1, 3, 4, 6}, {2, 5, 4, 6, 1, 3}, {2, 6, 1, 4, 3, 5}, {2, 6, 3, 5, 1, 4}, {3, 5, 1, 4, 2, 6}, {3, 5, 1, 6, 2, 4}, {3, 5, 2, 4, 1, 6}, {3, 5, 2, 6, 1, 4}, {4, 6, 1, 3, 2, 5}, {4, 6, 2, 5, 1, 3}. Since these are all in the form $\{a, b, c, d, e, f\}$, we will demonstrate in Example 4.1 how to convert the output into an actual crab canon sequence.

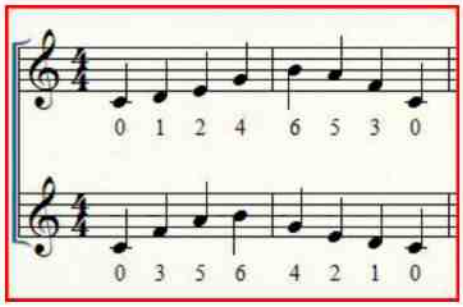
Example 4.1: Given {1, 3, 2, 5, 4, 6} in the form $\{a, b, c, d, e, f\}$, we know that a is the second note played, b is the seventh note played, etc. Then the order that these values will be played is actually {1, 2, 4, 6, 5, 3}. Adding on the first and last note we get the sequence {0, 1, 2, 4, 6, 5, 3, 0} which converted to notes is {C, D, E, G, B, A, F, C}. Table 4.3 shows how the Maple output converted into sequential order can be used to determine the notes played by Player 1 and Player 2.

Table 4.3

Player 1 (output forward)	0	1	2	4	6	5	3	0
Player 1(converted to notes)	C	D	E	G	B	A	F	C
Player 2(output backward)	0	3	5	6	4	2	1	0
Player 2 (converted to notes)	C	F	A	B	G	E	D	C
Interval Class	0	2	3	2	2	3	2	0

Analyzing the half-step intervals played we see that only the interval classes 0, 2, and 3 are being played in this crab canon sequence (See Table 4.3 above). Remember we restricted the use of interval classes in Algorithm #1 to 0, 2, 3, and 5.

With two voices for the crab canon, the result is the following line of music:



Observe this only creates a crab canon that is eight notes long. A longer crab canon sequence can easily be created by taking several eight note sequences and stringing them together.

4.4 Creating Algorithm #2

The output from Algorithm #1 gives us possible note sequences for a crab canon without dissonance. Yet upon listening we find that the first and last note being the same (C) does not create a very interesting melody. Since the first and last note are both C, the two voices are playing unison (interval class = 0) for both the first and last note of the crab canon sequence. We would like the first and last note of the crab canon sequence to be harmonious so this leads to a new idea used for Algorithm #2. We will set the rhythms played to be all of the same length, a quarter note, and let the notes be the seven tones from any major key; however now we will set the first and last note to be C and E which creates a major 3rd (interval class = 3). Note that based on our isomorphism definition it does not matter where the C and E are placed, i.e. {C, *a, c, e, f, d, b,* E} is equivalent to {E, *a, c, e, f, d, b,* C}. For the remaining six notes we will choose from all possible values 0 through 6 (C through B) which means that C or E (or both) may be used twice in the entire eight note sequence. In order to eliminate dissonance, we will still restrict the interval classes to 0, 3, 4, and 5; however the restrictions for Algorithm #2 will not allow for any unison so interval class 0 no longer needs to be considered.

We will use Table 4.4 for the enumeration of the 7 notes referenced in Algorithm #2.

Table 4.4	
Note	Value
C	1
D	2
E	3
F	4
G	5
A	6
B	7

Based on the interval class restrictions set for Algorithm #2, we get the following possible note pairings:

1→{3,4,5,6}

2→{4,5,6,7}

3→{1,5,7}

4→{1,2,6,7}

5→{1,2,3,7}

6→{1,2,4}

7→{2,3,4,5}

Based on the restrictions we have set, the crab canon will be an 8-note sequence with the first and last note being C (= 0) and E (= 2). Let the 8-note sequence be {0, *a, c, e, f, d, b*, 2} where 0 represents the note C, 2 represents the note E, *a* represents the second note played, *c* represents the third note played, etc. Then the intervals that will be played based on this sequence are (0, 2), (*a, b*), (*c, d*), and (*e, f*) and we want these intervals to belong to interval classes 3, 4, or 5. Based on Definition 3.1, the sequence {*a, c, e, f, d, b*} is equivalent to all of these sequences: {*b, c, e, f, d, a*}, {*a, d, e, f, c, b*}, {*a, c, f, e, d, b*}, {*b, d, e, f, c, a*}, {*b, c, f, e, d, a*}, {*a, d, f, e, c, b*}, and {*b, d, f, e, c, a*}.

Upon writing Algorithm #2, we noticed that we can eliminate isomorphisms from the Maple output by taking a closer look at how we are choosing the intervals (a, b), (c, d), and (e, f) in the algorithm. Since a and b are played together we know that (1, 3) is equivalent to (3, 1). Similarly, we know that (1,4) is equivalent to (4, 1), (1, 5) is equivalent to (5, 1), and so on. Then we can eliminate the isomorphism (3, 1) (as well as (4,1), (5,1), etc.) from our output by not allowing it to be chosen in the algorithm itself. We can let $a < b$ and $c < d$ in order to eliminate (3, 1) as a possible output in the algorithm. As a result, when we are choosing a and choosing c, we know that a will always be smaller than b and c will always be smaller than d.

By making adjustments to what we have for note pairings above, we now only have the following possible note pairings for Algorithm #2:

1→{3,4,5,6}

2→{4,5,6,7}

3→{5,7}

4→{6,7}

5→{7}

Notice that since all the possible note pairings for both 6 and 7 previously only included values less than the number itself, both 6 and 7 are eliminated as possible values for a and c. Accounting for the isomorphisms when choosing the last two notes, e and f, will be dealt with in the algorithm itself (see Section 4.5).

Here is an outline of Algorithm #2. The formal algorithm as written using Maple is in Appendix C.

4.5 Outline of Algorithm #2

Let the first and last note equal 1 and 3 (tonic and major third). We need to find all the possible ways to use 1 through 7 in positions "a - f" so that each number is only used once. "a"

represents the second note played, "b" represents the seventh note, "c" represents the third note, etc.

We need to find all the possible ways to use 1 through 6 in positions "a - f" so that each number is only used once. Remember a represents the second note played, b represents the seventh note, c represents the third note, etc.

Step 1: First choose a from $\{1,2,3,4,5\}$.

Step 2: If $a = 1$, choose b from $\{3,4,5,6\}$.

If $a = 2$, choose b from $\{4,5,6,7\}$.

If $a = 3$, choose b from $\{1,5,7\}$.

If $a = 4$, choose b from $\{1,2,6,7\}$.

If $a = 5$, choose b from $\{1,2,3,7\}$.

Step 3: Next choose c from $\{1,2,3,4,5\}$ but check to make sure that $c \neq a, c \neq b$.

Step 4: If $c = 1$, choose $d = 3$. If 3 has been used, choose $d = 4$. If 4 has been used, choose $d = 5$. If 5 has been used, choose $d = 6$.

If $c = 2$, choose $d = 4$. If 4 has been used, choose $d = 5$. If 5 has been used, choose $d = 6$. If 6 has been used, choose $d = 7$.

If $c = 3$, choose $d = 1$. If 1 has been used, choose $d = 5$. If 5 has been used, choose $d = 7$.

If $c = 4$, choose $d = 1$. If 1 has been used, choose $d = 2$. If 2 has been used, choose $d = 6$. If 6 has been used, choose $d = 7$.

If $c = 5$, choose $d = 1$. If 1 has been used, choose $d = 2$. If 2 has been used, choose $d = 3$. If 3 has been used, choose $d = 7$.

Step 5: There are three unused values left. Let $e < f < g$ and $e, f,$ and g cannot equal $a, b, c,$ or d.

Step 6: Check the values of $e, f,$ and g to find the last pair of notes.

Step 6a)

If e is from $\{1,2,3,4,5\}$ and $f - e = 2$, then choose e and f.

If e is from $\{1, 2, 4\}$ and $f - e = 3$, then choose e and f.

If e is from $\{1, 2, 3\}$ and $f \quad e = 4$, then choose e and f.

If e is from $\{1, 2\}$ and $f - e = 5$, then choose e and f.

Step 6b) Repeat Step 6a) replacing f with g.

Step 6c) Repeat Step 6a) replacing e with f and replacing f with g.

Repeat steps until all possible choices are used. If during Steps 2, 4, or 6 the result is not a possible note pairing, then throw it out and return to Step 1.

4.6 Results of Algorithm #2

The algorithm gave a total of 162 different sequences. For the full list see Appendix C. We created the algorithm in such a way that all sequences given in the output are unique so we do not need to account for isomorphisms as we did for Algorithm #1. We can use the output from Algorithm #2 to create an actual crab canon sequence that can be played using two voices the same way as outlined in Example 4.1.

After examining the output for Algorithm #2, the numbers of times each note pairing appears was analyzed.

Note	Algorithm Matrix can be played with	# of possible
C	E,F,G,A	4
D	F,G,A,B	4
E	G,A,B,C	4
F	D,A,C,	3
G	D,E,B,C	4
A	D,E,F,C	4
B	D,E,G	3

Single Notes in 6 string		
Note	appears #	doesn't appear #
C	144	18
D	144	18
E	144	18
F	132	30
G	138	24
A	138	24
B	132	24

notes compared to C in 6 note string		
Note	C or note	both C and note
D	36	126
E	36	126
F	48	114
G	42	120
A	42	120
B	48	114

See Appendix D for the full analysis. The analysis shows some sort of pattern or symmetry which seems to suggest that there may be a way to explain or predict the note pairings found in the algorithm. At this point no explanation or predicting factor has been found.

Chapter 5: Further Research

Throughout this thesis we have discussed the counting of certain melodic and rhythmic sequences (Chapter 1), the counting of asymmetrical rhythms (Chapter 2), the analysis of crab canons using interval class (Chapter 3), and we have written two algorithms that can be used to create crab canons without dissonance (Chapter 4). Further research exists and can be expanded in all of these areas.

Based on Rothstein [10] and Hook's [6] work, further research can be conducted on the connections between math and music, from the philosophical implications of mathematics and music both being art forms to the combinatorics perspective of counting the number of possible ways to create a musical composition satisfying certain constraints. Hook's article [6] goes into extensive detail on counting tetrachords, a pitch class set containing exactly four notes. Are there other interesting pitch class set relationships or can pitch class set be combined with another musical relationship to count another set of discrete possibilities?

The work done by Hall and Klingsberg [4,5] on asymmetrical rhythms (see Chapter 2) suggests many related questions on rhythms and canons that could be investigated further. Some topics that can be researched include producing a tiling canon of maximal category, exploring tiling canons with outer rhythms that are not equally spaced, rhythmic canons involving augmentation (where voices have rhythms in different meters), modifying tiling canons in which the rhythm patterns are inversions of each other, and finding a formula for the number of rhythms with the oddity property.

Vuza [11] analyzes tiling canons of maximal category. We saw in Chapter 2 that a tiling canon is a canon with the restriction that when all voices are played, the resultant rhythm has

exactly one note onset per unit beat. One with maximal category requires that both the inner and outer rhythms are primitive which means that rhythms do not appear in with a smaller period (i.e. no patter repeats within the period). Vuza shows that no nontrivial tiling canons of maximal category exist for periods less than 72, but there is no known formula for the number of tiling canons of maximal category. Is there a way to count tiling canons of maximal category given the length of the period?

We saw in Chapter 2 the work from Hall and Klingsberg provides a general formula that counts the number of tiling canons of l voices, offset by multiples of n notes for any ln-periodic rhythm with n note onsets. Their work is restricted to tiling canons offset by a multiple of n notes. In [7], Meyerowitz shows that any rhythm with three note onsets and the rhythm's inversion (playing the rhythm backwards) will always produce a tiling canon. For example, 110010 and 0011010 will produce a tiling canon when played together. What other values of n will produce tiling canons? Is there a general formula?

Other work concerns the oddity property. A rhythm has the rhythmic oddity property if it does not contain two onsets that partition the cycle into two half-cycles. In other words, there are no two note onsets that split the rhythm cycle in half. Chemillier[2,3] and Truchet[3] have developed an algorithm that gives all the rhythms formed from 2- or 3-unit notes having this property. Is there a general formula for finding rhythms with the oddity property?

In Chapter 3 of this thesis, we analyzed the interval classes used in Bach's "2 Canon *a 2*" from Musical Offering. This piece can be analyzed further based on its rhythms and perhaps other components. Also, other canon compositions can be analyzed and then a technique from David Cope's work (see [1]) can be modified to create a computer algorithm program that could compose canons similar to the musical compositions that have been analyzed. A less ambitious

approach would be to simply modify the algorithms given in Chapter 4 in some way to create a more musically well-rounded piece. Suggestions would include modifying the algorithm to allow for differing rhythm patterns and some dissonance that has resolution with it.

References

[1] Cope, David. (2001). *Virtual Music: Computer Synthesis of Musical Style*. Cambridge, Massuchessetts: The MIT Press.

[2] Chemillier, Marc. Ethnomusicaology, ethnomathematics. The logic underlying orally transmitted artistic practices, in *Mathematics and Music* (Lisbon/Vienna/Paris, 1999). pp. 161-183. Springer, Berlin, 2002.

[3] Chemillier, Marc and Truchet, Charlotte. Computation of words satisfying the rhythmic oddity property (after Simha Arom's works). *Information Processing Letters*, 86:255-261, 2003.

[4] Hall, R.W. and Klingsberg, K. Asymmetric rhythms, tiling canons, and Burnside's lemma, in *Bridges: Mathematical Connections in Art, Music, and Science*, R. Sarhangi and C. Sequin, eds., Winfield, KS, 2004, pp. 189-194.

[5] Hall, R. W. and Klingsberg, K. Asymmetric Rhythms and Tiling Canons, in *The American Mathematical Monthly*, Vol. 113, No. 10 (December, 2006), pp. 887-896.

[6] Hook, Julian. "Why Are There 29 Tetrachords? A Tutorial on Combinatorics and Enumeration in Music Theory." *Music Theory Online*, Volume 13, Number 4, December 2007. Retrieved April 6, 2012 from http://www.mtosmt.org/issues/mto.07.13.4/mto.07.13.4.hook.html#FN1.

[7] Meyerowitz. Tiling the line with triples, in *Discrete Models: combinatorics, Computation, and Geometry (Paris, 2001),* Discrete Math. Theor.Comput. Sci. Proc., AA, Mason Inform. Math. Discret. (MIMD), Paris, 2001, pp. 257-274 (electronic).

[8] Pitch Interval. (2012, January 12). In *Wikipedia*. Retrieved April 6, 2012, from http://en.wikipedia.org/wiki/Pitch_interval.

[9] Roman, Dan. "Twelve-Tone Technique: A Quick Reference." Retrieved April 6, 2012 from http://musike.cmpr.edu/v001/roman-eng.pdf.

[10] Rothstein, Edward. "The Inner Life of Music and Mathematics." *Math Horizons*. November 1995. pp. 6-8.

[11] Vuza, Dan Tudor. Supplementary sets and regular complementary unending canons I-IV. *Perspectives of New Music*, 29(2)-31(1), 1991-1993.

[12] Whittall, Arnold. *The Cambridge Introduction to Serialism*. Cambridge Introductions to Music. New York: Cambridge University Press. 2008.

Appendix A

Bach's Crab Canon Analysis where F = 0

Player 1	Player 2	interval in half steps	interval class
9	9	0	0
9	9	0	0
9	0	9	3
9	0	9	3
0	4	4	4
0	4	4	4
0	9	9	3
0	9	9	3
4	8	4	4
4	9	5	5
4	11	7	5
4	0	4	4
5	2	3	3
5	0	5	5
5	11	6	6
5	9	4	4
8	11	3	3
8	4	4	4
8	11	3	3
8	2	6	6
	0	0	
	11	11	
4	9	5	5
4	8	4	4
4	6	2	2
4	8	4	4
3	11	8	4
3	0	3	3
3	11	8	4
3	9	6	6
2	8	6	6
2	6	4	4
2	4	2	2
2	5	3	3
1	7	6	6
1	10	9	3

1	9	8	4
1	7	6	6
0	5	5	5
0	4	4	4
0	2	2	2
0	4	4	4
11	5	6	6
11	7	4	4
10	5	5	5
10	4	6	6
9	2	7	5
9	0	9	3
8	11	3	3
8	0	8	4
4	2	2	2
4	4	0	0
9	2	7	5
9	0	9	3
2	11	9	3
2	5	3	3
0	4	4	4
0	2	2	2
0	0	0	0
0	9	9	3
11	8	3	3
11	6	5	5
11	4	7	5
11	2	9	3
9	0	9	3
9	11	2	2
9	0	9	3
9	4	5	5
0	9	9	3
0	4	4	4
0	2	2	2
0	4	4	4
4	0	4	4
2	0	2	2
4	0	4	4
9	0	9	3

4	9	5	5
0	9	9	3
11	9	2	2
0	9	9	3
2	11	9	3
4	11	7	5
6	11	5	5
8	11	3	3
9	0	9	3
0	0	0	0
2	0	2	2
4	0	4	4
5	2	3	3
11	2	9	3
0	9	9	3
2	9	7	4
4	4	0	0
2	4	2	2
0	8	8	4
11	8	3	3
0	9	9	3
2	9	7	5
4	10	6	6
5	10	5	5
7	11	4	4
5	11	6	6
4	0	4	4
2	0	2	2
4	0	4	4
5	0	5	5
7	1	6	6
9	1	8	4
10	1	9	3
7	1	6	6
5	2	3	3
4	2	2	2
6	2	4	4
8	2	6	6
9	3	6	6
11	3	8	4
0	3	3	3

11	3	8	4
8	4	4	4
6	4	2	2
8	4	4	4
9	4	5	5
11		11	
0		0	
2	8	6	6
11	8	3	3
4	8	4	4
11	8	3	3
9	5	4	4
11	5	6	6
0	5	5	5
2	5	3	3
0	4	4	4
11	4	7	5
9	4	5	5
8	4	4	4
9	0	9	3
9	0	9	3
4	0	4	4
4	0	4	4
0	9	9	3
0	9	9	3
9	9	0	0
9	9	0	0

Appendix B

Marissa's Crab Canon Algorithm #1

$$Aposs := \begin{bmatrix} 3 & 4 & 5 & 6 & 0 \\ 4 & 5 & 6 & 0 & 0 \\ 1 & 5 & 0 & 0 & 0 \\ 1 & 2 & 6 & 0 & 0 \\ 1 & 2 & 3 & 0 & 0 \\ 1 & 2 & 4 & 0 & 0 \end{bmatrix} :$$

$$Cposs := \begin{bmatrix} 3 & 4 & 5 & 6 & 0 \\ 4 & 5 & 6 & 0 & 0 \\ 1 & 5 & 0 & 0 & 0 \\ 1 & 2 & 6 & 0 & 0 \\ 1 & 2 & 3 & 0 & 0 \\ 1 & 2 & 4 & 0 & 0 \end{bmatrix} :$$

$cntr := 0 :$

```
for a from 1 to 6
 do
  used := Array(1..6, [0, 0, 0, 0, 0, 0]) :
  used[a] := 1 :
  acntr := 1 :
  while Aposs[a, acntr] ≠ 0
   do
    b := Aposs[a, acntr] :
    used[b] := 1 :
    for c from 1 to 6
     do used[a] := 1 : used[b] := 1 :
      if used[c] = 0
       then ccntr := 1 :
        used[c] := 1 :
         while Cposs[c, ccntr] ≠ 0
          do
           d := Cposs[c, ccntr] : #print(a, b, c, d) :
           if used[d] = 1
            then ccntr := ccntr + 1
            else used[d] := 1 :
                 e := 1 :
                  while used[e] = 1 do e := e + 1 od:
                  f := e + 1 :
                  while used[f] = 1 do f := f + 1 od:
```

```
                    if e in {1, 2, 3, 4}
                     then if f − e = 2 then print(a, b, c, d, e, f) : print(a, b, c, d, f, e) : cntr := cntr
 + 2 :
                                        else if e in {1, 2} and f − e in {3, 4} then print(a, b, c, d,
e, f) : print(a, b, c, d, f, e) : cntr := cntr + 2 :
                                                                         else if e = 1
and f − e = 5 then print(a, b, c, d, e, f) : print(a, b, c, d, f, e) : cntr := cntr + 2 :
                                                                                 fi:
                                             fi:
                          fi:
                   fi:
                #    if |e − f| in {3, 4, 5, 7, 8, 9} then print(a, b, c, d, e, f) : fi:
             #       temp:=e: e:=f: f:=temp:
             #       if |e − f| in {3, 4, 5, 7, 8, 9} then print(a, b, c, d, e, f) fi:
```

```
                  used[d] := 0 :
                   ccntr := ccntr + 1 :
             fi:
          od:
  #closes off the "while Cposs[c,ccntr]" loop; ccntr has been incremented; look at the next value in
  the same row of Cposs.
        else ccntr := 0 : #This indicates the value of c currently set was not used.
        if ccntr ≠ 0 then used[c] = 0 fi:
  #Only want to reset used[c] if it doesn't coincide with used[a] or used[b].
      fi: #closes off the "if used[c]=0" loop; pursued this level of a,b,c to end.
       used[c] := 0 : #reset the value of used[c] as 0, as we're not using this value for c anymore.
     od:  #closes off the "for c from 1 to 6" loop; c now goes on to the next value for b.
  acntr := acntr + 1 :
  used[b] := 0 :
 od:
  #closes off the "while Aposs[a,acntr]" loop; have tried all possible matches with current value of a.
 used[a] := 0 : #all values of "used" should now be reset
od: #hits this when has run through all values of a.
```

```
1, 3, 2, 5, 4, 6
1, 3, 2, 5, 6, 4
1, 3, 4, 6, 2, 5
1, 3, 4, 6, 5, 2
1, 3, 5, 2, 4, 6
1, 3, 5, 2, 6, 4
1, 3, 6, 4, 2, 5
1, 3, 6, 4, 5, 2
1, 4, 2, 6, 3, 5
1, 4, 2, 6, 5, 3
```

1, 4, 3, 5, 2, 6
1, 4, 3, 5, 6, 2
1, 4, 5, 3, 2, 6
1, 4, 5, 3, 6, 2
1, 4, 6, 2, 3, 5
1, 4, 6, 2, 5, 3
1, 6, 2, 4, 3, 5
1, 6, 2, 4, 5, 3
1, 6, 3, 5, 2, 4
1, 6, 3, 5, 4, 2
1, 6, 4, 2, 3, 5
1, 6, 4, 2, 5, 3
1, 6, 5, 3, 2, 4
1, 6, 5, 3, 4, 2
2, 4, 1, 6, 3, 5
2, 4, 1, 6, 5, 3
2, 4, 3, 5, 1, 6
2, 4, 3, 5, 6, 1
2, 4, 5, 3, 1, 6
2, 4, 5, 3, 6, 1
2, 4, 6, 1, 3, 5
2, 4, 6, 1, 5, 3
2, 5, 1, 3, 4, 6
2, 5, 1, 3, 6, 4
2, 5, 3, 1, 4, 6
2, 5, 3, 1, 6, 4
2, 5, 4, 6, 1, 3

2, 5, 4, 6, 3, 1
2, 5, 6, 4, 1, 3
2, 5, 6, 4, 3, 1
2, 6, 1, 4, 3, 5
2, 6, 1, 4, 5, 3
2, 6, 3, 5, 1, 4
2, 6, 3, 5, 4, 1
2, 6, 4, 1, 3, 5
2, 6, 4, 1, 5, 3
2, 6, 5, 3, 1, 4
2, 6, 5, 3, 4, 1

3, 1, 2, 5, 4, 6
3, 1, 2, 5, 6, 4
3, 1, 4, 6, 2, 5
3, 1, 4, 6, 5, 2
3, 1, 5, 2, 4, 6
3, 1, 5, 2, 6, 4
3, 1, 6, 4, 2, 5
3, 1, 6, 4, 5, 2
3, 5, 1, 4, 2, 6
3, 5, 1, 4, 6, 2
3, 5, 1, 6, 2, 4
3, 5, 1, 6, 4, 2
3, 5, 2, 4, 1, 6
3, 5, 2, 4, 6, 1
3, 5, 2, 6, 1, 4
3, 5, 2, 6, 4, 1
3, 5, 4, 1, 2, 6
3, 5, 4, 1, 6, 2
3, 5, 4, 2, 1, 6
3, 5, 4, 2, 6, 1
3, 5, 6, 1, 2, 4
3, 5, 6, 1, 4, 2
3, 5, 6, 2, 1, 4
3, 5, 6, 2, 4, 1
4, 1, 2, 6, 3, 5
4, 1, 2, 6, 5, 3
4, 1, 3, 5, 2, 6
4, 1, 3, 5, 6, 2

4, 1, 5, 3, 2, 6
4, 1, 5, 3, 6, 2
4, 1, 6, 2, 3, 5
4, 1, 6, 2, 5, 3
4, 2, 1, 6, 3, 5
4, 2, 1, 6, 5, 3
4, 2, 3, 5, 1, 6
4, 2, 3, 5, 6, 1
4, 2, 5, 3, 1, 6
4, 2, 5, 3, 6, 1

4, 2, 6, 1, 3, 5
4, 2, 6, 1, 5, 3
4, 6, 1, 3, 2, 5
4, 6, 1, 3, 5, 2
4, 6, 2, 5, 1, 3
4, 6, 2, 5, 3, 1
4, 6, 3, 1, 2, 5
4, 6, 3, 1, 5, 2
4, 6, 5, 2, 1, 3
4, 6, 5, 2, 3, 1
5, 2, 1, 3, 4, 6
5, 2, 1, 3, 6, 4
5, 2, 3, 1, 4, 6
5, 2, 3, 1, 6, 4
5, 2, 4, 6, 1, 3
5, 2, 4, 6, 3, 1
5, 2, 6, 4, 1, 3
5, 2, 6, 4, 3, 1
5, 3, 1, 4, 2, 6
5, 3, 1, 4, 6, 2
5, 3, 1, 6, 2, 4
5, 3, 1, 6, 4, 2
5, 3, 2, 4, 1, 6
5, 3, 2, 4, 6, 1
5, 3, 2, 6, 1, 4
5, 3, 2, 6, 4, 1
5, 3, 4, 1, 2, 6
5, 3, 4, 1, 6, 2

5, 3, 4, 2, 1, 6
5, 3, 4, 2, 6, 1
5, 3, 6, 1, 2, 4
5, 3, 6, 1, 4, 2
5, 3, 6, 2, 1, 4
5, 3, 6, 2, 4, 1
6, 1, 2, 4, 3, 5
6, 1, 2, 4, 5, 3
6, 1, 3, 5, 2, 4
6, 1, 3, 5, 4, 2

6, 1, 4, 2, 3, 5
6, 1, 4, 2, 5, 3
6, 1, 5, 3, 2, 4
6, 1, 5, 3, 4, 2
6, 2, 1, 4, 3, 5
6, 2, 1, 4, 5, 3
6, 2, 3, 5, 1, 4
6, 2, 3, 5, 4, 1
6, 2, 4, 1, 3, 5
6, 2, 4, 1, 5, 3
6, 2, 5, 3, 1, 4
6, 2, 5, 3, 4, 1
6, 4, 1, 3, 2, 5
6, 4, 1, 3, 5, 2
6, 4, 2, 5, 1, 3
6, 4, 2, 5, 3, 1
6, 4, 3, 1, 2, 5
6, 4, 3, 1, 5, 2
6, 4, 5, 2, 1, 3
6, 4, 5, 2, 3, 1

cntr

144 **(2)**

Appendix C

Marissa's Crab Canon Algorithm #2

$$Aposs := \begin{bmatrix} 3 & 4 & 5 & 6 & 0 \\ 4 & 5 & 6 & 7 & 0 \\ 5 & 7 & 0 & 0 & 0 \\ 6 & 7 & 0 & 0 & 0 \\ 7 & 0 & 0 & 0 & 0 \end{bmatrix} :$$

$cntr := 0 :$

```
for a from 1 to 5
 do
  used := Array(1..7, [0, 0, 0, 0, 0, 0, 0]) : #print("ok") :
  used[a] := 1 :
  acntr := 1 :
  while Aposs[a, acntr] ≠ 0
   do
    b := Aposs[a, acntr] : #print("OK 2", a, b) :
    used[b] := 1 : #print(a, b) :
    for c from 1 to 5
     do used[a] := 1 : used[b] := 1 :
      if used[c] = 0
       then ccntr := 1 :
        used[c] := 1 :
         while Aposs[c, ccntr] ≠ 0
          do
           d := Aposs[c, ccntr] : #print(a, b, c, d) :
           if used[d] = 1
            then ccntr := ccntr + 1
            else used[d] := 1 :
                 e := 1 :
                  while used[e] = 1 do e := e + 1 od:
                  f := e + 1 :
                  while used[f] = 1 do f := f + 1 od:
                   g := f + 1 :
                  while used[g] = 1 do g := g + 1 od:
                 #FINDING THE LAST PAIR E AND F
                  if e in {1, 2, 3, 4, 5} and f - e = 2 then print(a, b, c, d, e, f) : cntr := cntr + 1 :
 fi:
                  if e in {1, 2, 4} and f - e = 3 then print(a, b, c, d, e, f) : cntr := cntr + 1 : fi :
 #i'm counting 2 even though I am only printing one of the two (they are isomorphic)
                  if e in {1, 2, 3} and f - e = 4 then print(a, b, c, d, e, f) : cntr := cntr + 1 : fi:
                  if e in {1, 2} and f - e = 5 then print(a, b, c, d, e, f) : cntr := cntr + 1 :  fi:
                  #FINDING THE LAST PAIR E AND G
```

```
                if e in {1, 2, 3, 4, 5} and g − e = 2 then print(a, b, c, d, e, g) : cntr := cntr + 1 :
fi:
                if e in {1, 2, 4} and g − e = 3 then print(a, b, c, d, e, g) : cntr := cntr + 1 : fi :
#i'm counting 2 even though I am only printing one of the two (they are isomorphic)
                if e in {1, 2, 3} and g − e = 4 then print(a, b, c, d, e, g) : cntr := cntr + 1 : fi:
                if e in {1, 2} and g − e = 5 then print(a, b, c, d, e, g) : cntr := cntr + 1 :  fi:
               #FINDING THE LAST PAIR F AND G
                if f in {1, 2, 3, 4, 5} and g − f = 2 then print(a, b, c, d, f, g) : cntr := cntr + 1 :
fi:
                if f in {1, 2, 4} and g − f = 3 then print(a, b, c, d, f, g) : cntr := cntr + 1 : fi :
#i'm counting 2 even though I am only printing one of the two (they are isomorphic)
                if f in {1, 2, 3} and g − f = 4 then print(a, b, c, d, f, g) : cntr := cntr + 1 : fi:
                if f in {1, 2} and g − f = 5 then print(a, b, c, d, f, g) : cntr := cntr + 1 :  fi:
```

```
                used[d] := 0 :
                 ccntr := ccntr + 1 :
            fi:
        od:
    #closes off the "while Cposs[c,ccntr]" loop; ccntr has been incremented; look at the next value in
    the same row of Cposs.
          else ccntr := 0 : #This indicates the value of c currently set was not used.
          if ccntr ≠ 0 then used[c] = 0 fi:
    #Only want to reset used[c] if it doesn't coincide with used[a] or used[b].
        fi: #closes off the "if used[c]=0" loop; pursued this level of a,b,c to end.
         used[c] := 0 : #reset the value of used[c] as 0, as we're not using this value for c anymore.
       od:  #closes off the "for c from 1 to 6" loop; c now goes on to the next value for b.
     acntr := acntr + 1 :
     used[b] := 0 :
   od:
     #closes off the "while Aposs[a,acntr]" loop; have tried all possible matches with current value of a.
  used[a] := 0 : #all values of "used" should now be reset
od: #hits this when has run through all values of a.
```

1, 3, 2, 4, 5, 7
1, 3, 2, 5, 4, 6
1, 3, 2, 5, 4, 7
1, 3, 2, 6, 4, 7
1, 3, 2, 6, 5, 7
1, 3, 2, 7, 4, 6
1, 3, 4, 6, 2, 5
1, 3, 4, 6, 2, 7
1, 3, 4, 6, 5, 7
1, 3, 4, 7, 2, 5
1, 3, 4, 7, 2, 6
1, 3, 5, 7, 2, 4

1, 3, 5, 7, 2, 6
1, 3, 5, 7, 4, 6
1, 4, 2, 5, 3, 7
1, 4, 2, 6, 3, 5
1, 4, 2, 6, 3, 7
1, 4, 2, 6, 5, 7
1, 4, 2, 7, 3, 5
1, 4, 3, 5, 2, 6
1, 4, 3, 5, 2, 7
1, 4, 3, 7, 2, 5
1, 4, 3, 7, 2, 6
1, 4, 5, 7, 2, 6
1, 5, 2, 4, 3, 7
1, 5, 2, 6, 3, 7
1, 5, 2, 6, 4, 7
1, 5, 2, 7, 4, 6
1, 5, 3, 7, 2, 4
1, 5, 3, 7, 2, 6
1, 5, 3, 7, 4, 6
1, 5, 4, 6, 2, 7
1, 5, 4, 6, 3, 7
1, 5, 4, 7, 2, 6
1, 6, 2, 4, 3, 5
1, 6, 2, 4, 3, 7
1, 6, 2, 4, 5, 7
1, 6, 2, 5, 3, 7
1, 6, 2, 5, 4, 7
1, 6, 2, 7, 3, 5
1, 6, 3, 5, 2, 4
1, 6, 3, 5, 2, 7
1, 6, 3, 5, 4, 7
1, 6, 3, 7, 2, 4
1, 6, 3, 7, 2, 5
1, 6, 4, 7, 2, 5
1, 6, 4, 7, 3, 5
1, 6, 5, 7, 2, 4
2, 4, 1, 3, 5, 7
2, 4, 1, 5, 3, 7

2, 4, 1, 6, 3, 5
2, 4, 1, 6, 3, 7
2, 4, 1, 6, 5, 7
2, 4, 3, 5, 1, 6
2, 4, 3, 7, 1, 5
2, 4, 3, 7, 1, 6
2, 4, 5, 7, 1, 3
2, 4, 5, 7, 1, 6
2, 5, 1, 3, 4, 6
2, 5, 1, 3, 4, 7
2, 5, 1, 4, 3, 7
2, 5, 1, 6, 3, 7
2, 5, 1, 6, 4, 7
2, 5, 3, 7, 1, 4
2, 5, 3, 7, 1, 6
2, 5, 3, 7, 4, 6
2, 5, 4, 6, 1, 3
2, 5, 4, 6, 3, 7
2, 5, 4, 7, 1, 3
2, 5, 4, 7, 1, 6
2, 6, 1, 3, 4, 7
2, 6, 1, 3, 5, 7
2, 6, 1, 4, 3, 5
2, 6, 1, 4, 3, 7
2, 6, 1, 4, 5, 7
2, 6, 1, 5, 3, 7
2, 6, 1, 5, 4, 7
2, 6, 3, 5, 1, 4
2, 6, 3, 5, 4, 7
2, 6, 3, 7, 1, 4
2, 6, 3, 7, 1, 5
2, 6, 4, 7, 1, 3
2, 6, 4, 7, 1, 5
2, 6, 4, 7, 3, 5
2, 6, 5, 7, 1, 3
2, 6, 5, 7, 1, 4
2, 7, 1, 3, 4, 6
2, 7, 1, 4, 3, 5

2, 7, 1, 5, 4, 6
2, 7, 1, 6, 3, 5
2, 7, 3, 5, 1, 4
2, 7, 3, 5, 1, 6
2, 7, 3, 5, 4, 6
2, 7, 4, 6, 1, 3
2, 7, 4, 6, 1, 5
2, 7, 4, 6, 3, 5
3, 5, 1, 4, 2, 6
3, 5, 1, 4, 2, 7
3, 5, 1, 6, 2, 4
3, 5, 1, 6, 2, 7
3, 5, 1, 6, 4, 7
3, 5, 2, 4, 1, 6
3, 5, 2, 6, 1, 4
3, 5, 2, 6, 4, 7
3, 5, 2, 7, 1, 4
3, 5, 2, 7, 1, 6
3, 5, 2, 7, 4, 6
3, 5, 4, 6, 2, 7
3, 5, 4, 7, 1, 6
3, 5, 4, 7, 2, 6
3, 7, 1, 4, 2, 5
3, 7, 1, 4, 2, 6
3, 7, 1, 5, 2, 4
3, 7, 1, 5, 2, 6
3, 7, 1, 5, 4, 6
3, 7, 1, 6, 2, 4
3, 7, 1, 6, 2, 5
3, 7, 2, 4, 1, 5
3, 7, 2, 4, 1, 6
3, 7, 2, 5, 1, 4
3, 7, 2, 5, 1, 6
3, 7, 2, 5, 4, 6
3, 7, 2, 6, 1, 4
3, 7, 2, 6, 1, 5
3, 7, 4, 6, 1, 5
3, 7, 4, 6, 2, 5

4, 6, 1, 3, 2, 5
4, 6, 1, 3, 2, 7
4, 6, 1, 3, 5, 7
4, 6, 1, 5, 2, 7
4, 6, 1, 5, 3, 7
4, 6, 2, 5, 1, 3
4, 6, 2, 5, 3, 7
4, 6, 2, 7, 1, 3
4, 6, 2, 7, 1, 5
4, 6, 2, 7, 3, 5
4, 6, 3, 5, 2, 7
4, 6, 3, 7, 1, 5
4, 6, 3, 7, 2, 5
4, 6, 5, 7, 1, 3
4, 7, 1, 3, 2, 5
4, 7, 1, 3, 2, 6
4, 7, 1, 5, 2, 6
4, 7, 1, 6, 2, 5
4, 7, 1, 6, 3, 5
4, 7, 2, 5, 1, 3
4, 7, 2, 5, 1, 6
4, 7, 2, 6, 1, 3
4, 7, 2, 6, 1, 5
4, 7, 2, 6, 3, 5
4, 7, 3, 5, 1, 6
4, 7, 3, 5, 2, 6

5, 7, 1, 3, 2, 4
5, 7, 1, 3, 2, 6
5, 7, 1, 3, 4, 6
5, 7, 1, 4, 2, 6
5, 7, 1, 6, 2, 4
5, 7, 2, 4, 1, 3
5, 7, 2, 4, 1, 6
5, 7, 2, 6, 1, 3
5, 7, 2, 6, 1, 4
5, 7, 4, 6, 1, 3

cntr

162 **(2)**

Appendix D

Note	Algorithm Matrix can be played with	# of possible
C	E,F,G,A	4
D	F,G,A,B	4
E	G,A,B,C	4
F	D,A,C,	3
G	D,E,B,C	4
A	D,E,F,C	4
B	D,E,G	3

Note	Single Notes in 6 string appears #	doesn't appear #
C	144	18
D	144	18
E	144	18
F	132	30
G	138	24
A	138	24
B	132	24

notes compared to C in 6 note string		
Note	C or note	both C and note
D	36	126
E	36	126
F	48	114
G	42	120
A	42	120
B	48	114

notes compared to D in 6 note string		
Note	D or note	both D and note
C	36	126
E	36	126
F	48	114
G	42	120
A	42	120
B	48	114

notes compared to E in 6 note string		
note	E or note	both E and note
C	36	126
D	36	126
F	48	114
G	42	120
A	42	120
B	48	114

notes compared to F in 6 note string		
note	F or note	both F and note
C	48	114
D	48	114
E	48	114
G	54	108
A	54	108
B	60	102

notes compared to G in 6 note string		
note	G or note	both G and note
C	42	120
D	42	120
E	42	120
F	54	108
A	48	114
B	54	108

notes compared to A in 6 note string		
note	A or note	both A and note
C	42	120
D	42	120
E	42	120
F	54	108
G	48	114
B	54	108

notes compared to B in 6 note string		
note	B or note	both B and note
C	48	114
D	48	114
E	48	114
F	60	102
G	54	108
A	54	108