$14.95

THE MUSICAL ATARI

by Hal Glicksman
Music arranged by Laura Goodfriend

Turn your Atari Home Computer into a keyboard organ!

THE MUSICAL ATARI

THE MUSICAL ATARI

by
Hal Glicksman

Music arranged by
Laura Goodfriend

Illustrated by
Martin Cannon

DATAMOST™ INC
20660 Nordhoff St.
Chatsworth, CA 91311-6152
(818) 709-1202

ISBN 0-88190-345-0

Printed in U.S.A.

ACKNOWLEDGMENTS


o my son, John, for giving me a reason to learn about computers. The computer has made an important contribution to both his life and mine. Laura Goodfriend, who did the arrangements for this book, is also his music teacher, so John is also responsible for the collaboration that resulted in this book.

To Laura Goodfriend who not only wrote wonderful songs for the computer, but taught me about music and helped with the text.

To my wife Mary Ann who was very patient while I was closeted away with the Atari.

To Dave Gordon for his many years of generosity and help as well as his encouragement to be creative.

To Marcia Carrozzo of the Book Division of DATAMOST, and Karen Williams, an excellent editor and a friend.

To Martin Cannon, a real artist.

Hal Glicksman

TABLE OF CONTENTS

PREFACE

e have tried to peer into a crystal ball, but were unable to foresee if you, our reader, already knew the secrets of programming in BASIC, or could sight-read a piece of sheet music on the kazoo. Maybe you can do both already, but we acted on the assumption that your computer is barely warmed up and that you have no prior knowledge of music.

We have endeavored to include:

- Simple computer programs that beginners can readily understand. These programs will allow you to use your Atari as a chord organ or a player piano.
- A basic introduction to music that can be applied to the piano as well as the Atari.
- A collection of popular songs arranged for both the Atari and the piano.

When you begin to play a musical instrument you have to memorize the music or be able to sight read at the right tempo for the music to sound the way it should. The joy of Atari music is that you can look up one note at a time, store the note in the computer, and play the music back at any tempo. The music can be stored on tape or disk and becomes part of your repertoire. You can change any note or group of notes at any time and store variations of the songs in the computer's memory.

Because computers are constantly changing and growing in capability, we have tried to present principles and methods that apply to other computers and other musical instruments. Once you have learned to read the songs in this book you can sing the songs or play them on the piano in addition to playing them on the Atari. The principles of BASIC programming which are introduced in the section on player programs will also help you with other programs and computer games. A chapter on sound effects gives you ideas for improving games and adventure programs.

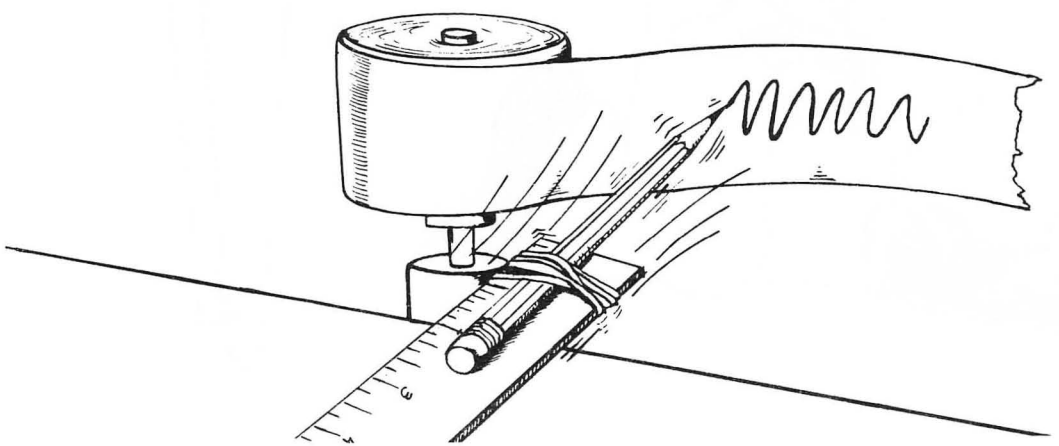
Please Feel Free

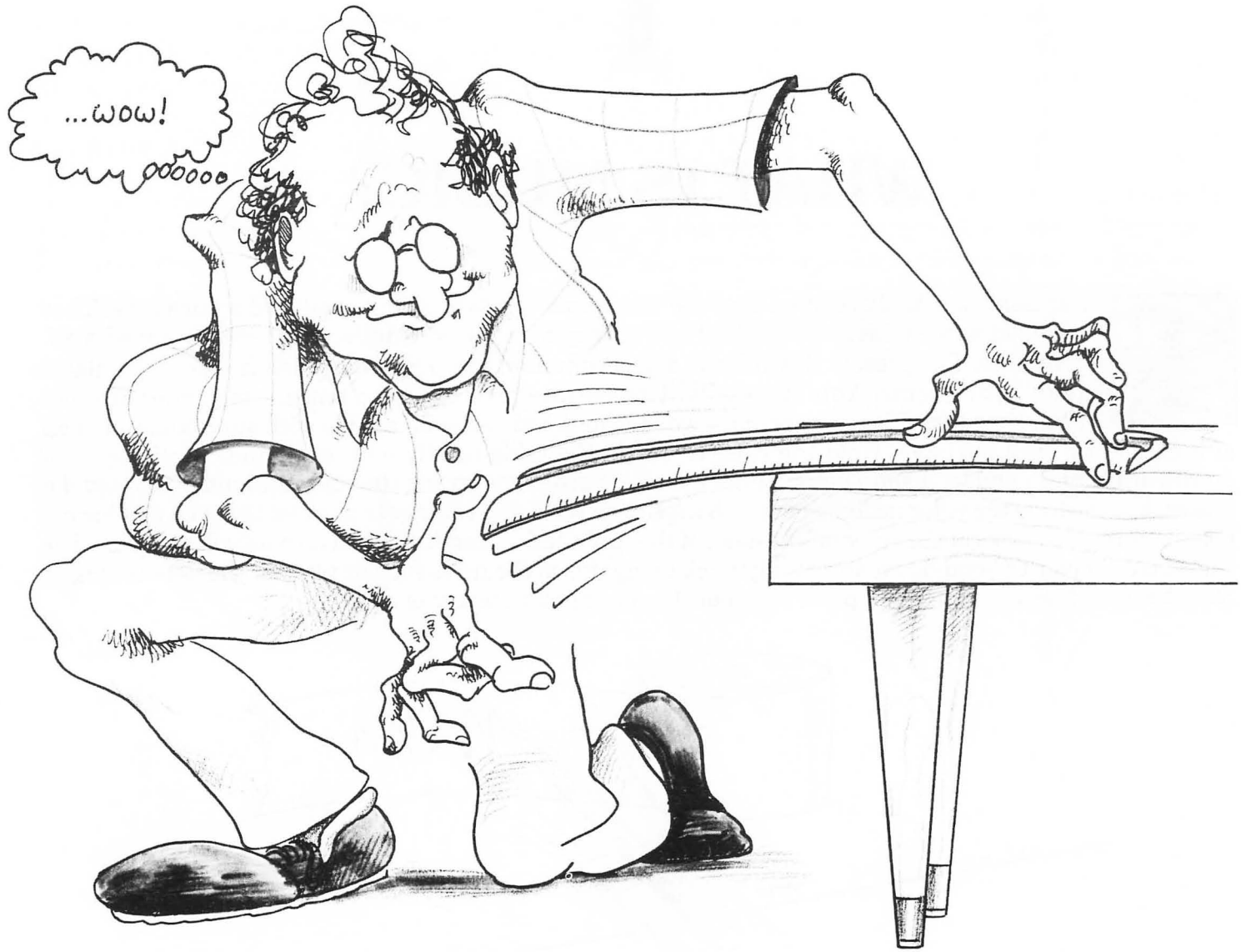
Some people are afraid to tinker with a recipe if it's printed in a cookbook. Don't be afraid. The programs in this book are like recipes, not like instructions on assembling an engine. Twenty different programmers would write twenty different player piano programs and they all would work. The songs, too, are a matter of taste; change anything you like. Get any piano song book and start writing Atari versions of your favorite songs. We are interested in unusual and creative applications of the player programs. Send us a copy of your music disk in care of DATAMOST, if you like.

1

WHAT IS MUSIC?

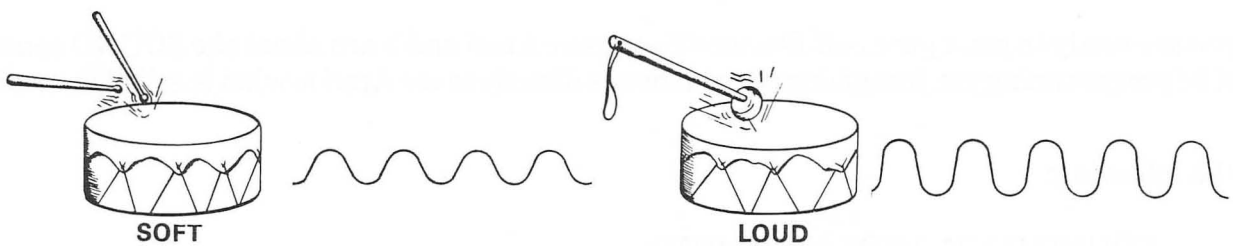
The ancient Greeks discovered that the vibrations of a bow string produced a pure tone. They developed musical instruments that had strings of different lengths. Each string played a different note. The present day harp is a descendant of the Greek harp and is one of the oldest known instruments. You can easily demonstrate how different tones come from different length vibrators. Place an ordinary wooden or plastic ruler on a table with approximately half of it extending over the edge. Hold the ruler down on the table tightly with one hand. Bend the ruler downward slightly and let it snap back. Notice that the harder you twang the ruler the louder the sound it makes, but the tone the ruler makes doesn't change even when the ruler gets so quiet that you can hardly hear it. Now stick more of the ruler off the edge of the table and twang it again. The note will be lower. The ruler takes longer to bend down and spring back up again, so the frequency or pitch of the note is longer. Still the note does not change in pitch whether it is twanged strongly or lightly.



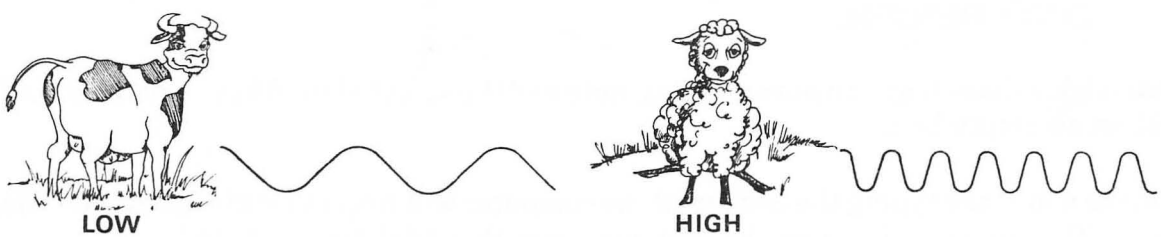

...wow!
oooooo

Imagine that our ruler has a pencil on the end, and as it vibrates we turn a big paper drum around to draw out the vibrations on a paper roll. The lower notes move the pencil up and down more slowly, so the hills and valleys on the paper are farther apart. Loud notes draw high hills and deep valleys, but the distance from hill to hill remains the same. The grooves in a phonograph record are very similar to the wiggly lines that our vibrating pencil draws on the paper. If you have an old 78 rpm record at home, it is very easy to see the waves in the grooves of the record with a magnifying glass. When the sound waves are even and smooth we call what we hear a pure tone or a pure note. When the sound wave consists of many different frequencies and volumes we have what is called noise. The tones that are used to make music are mainly pure tones, but the purity of the note is modified by the nature of the instrument that produces the tone. Another important ingredient in music is the organization of the notes in time. We will be learning how the computer produces tones and keeps time, and we will translate musical notes into "data," which is information that the computer can understand. This book will give you a few more principles of music as you go along, so that what you learn on the Atari will help you understand how other musical instruments work and show you how to read musical notation. Since words like "pitch" and "note" can be confusing, we have provided a glossary of musical terms at the back of the book.

AMPLITUDE



FREQUENCY



How Does the Atari Make Musical Notes?

Deep inside the Atari is a sound-generating microchip that produces very pure vibrations similar to those our pencil drew. The tone is purer than the note of a piano or other musical instrument. Initially, we will use this chip to make sounds and then music by sending commands to the Atari in BASIC computer language. This book will also teach you enough about BASIC to write a program which plays music, but not by using so much computer jargon that you'll get a headache.

Getting Started

Your Atari should be hooked up and the BASIC COMPUTING LANGUAGE cartridge should be inserted the cartridge slot. If your computer is brand new, take a little time with the instruction manual or a good beginner's book like *Kids and the Atari* or *The Elementary Atari.* We assume that the computer is working and you are looking at a nice clean screen with the word READY in the upper left corner. If you have other words or images on the screen remaining from other BASIC programs, clear the screen by pressing <CTRL> and <CLEAR> at the same time. Then type NEW and press <RETURN> to clear out any old memories and cobwebs.

Now you are ready to get a pure bell-like tone from your Atari and learn about the SOUND command. We will not be programming yet, just giving the commands directly to the Atari in what is called Immediate mode.

Type the following:

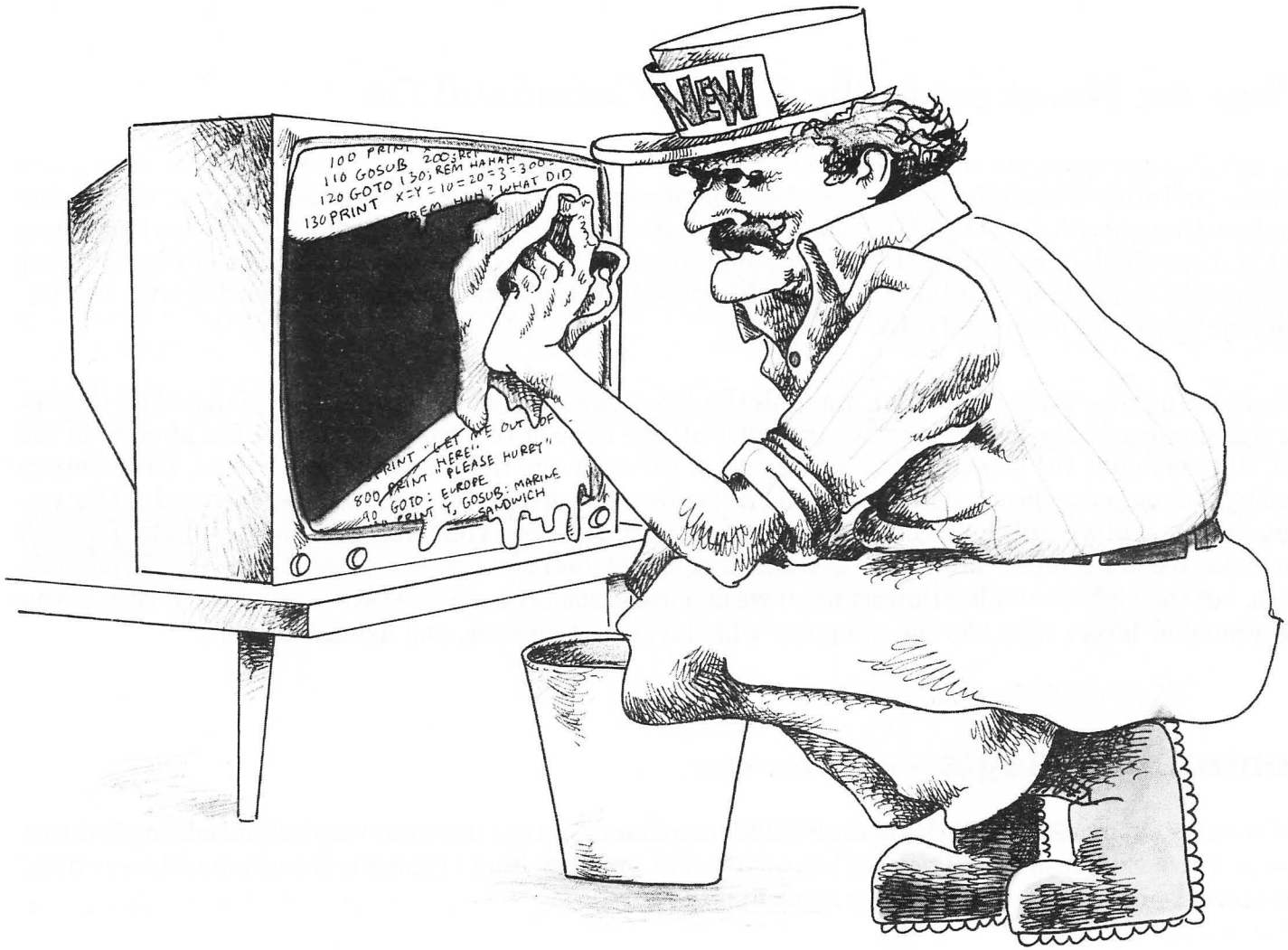
```
SOUND 0,121,10,08 <RETURN>
```

When you have the sound command working type:

```
END <RETURN>
```

The Atari should have played a pure middle C note and the screen should have printed READY. The cursor will rest on an empty line.

If you made a mistake typing the command the computer will respond with "ERROR--" and repeat what you typed. The cursor will rest on the first character that "did not compute."


NEW
120 GOTO 130; REM HAHA!
130 PRINT X=Y=10=20=3=300
PRINT "HERE"
PLEASE HURT"
GOTO: EUROPE
Y, GOSUB: MARINE
SANDWICH

This will shut off the sound and give us a chance to explain what just happened. The SOUND command was "executed" when we pressed <RETURN>, and the computer played the sound until directed to stop by the END command.

What the Numbers in the Sound Command Do

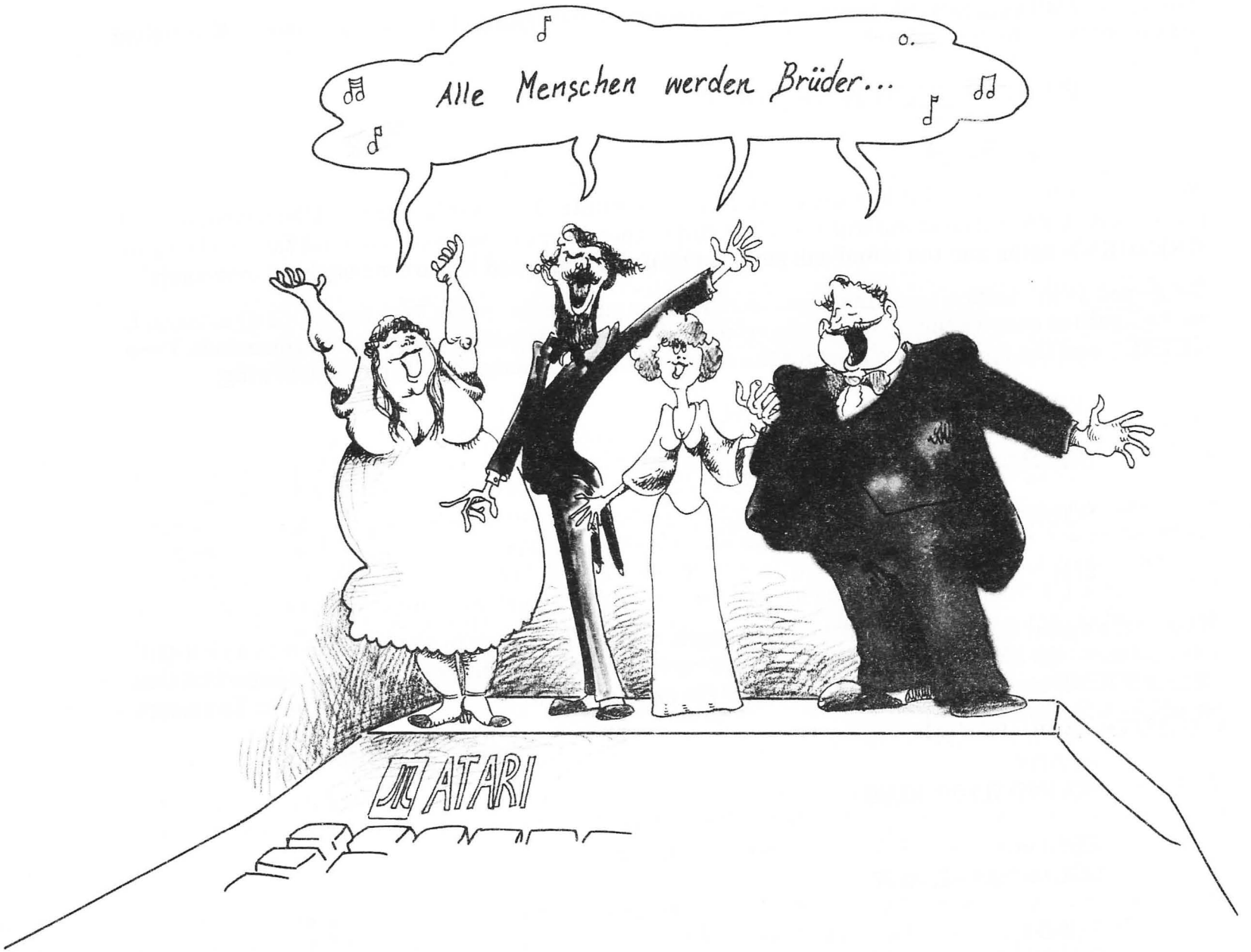
The SOUND command not only creates musical notes, but it can imitate explosions, waterfalls, sirens, bird whistles and other sound effects for which Atari games are famous. All of these effects are controlled by the numbers that go with the SOUND command. The first number selects the **voice**. The Atari has four voices that you can think of as a quartet in music: four singers, four violins, or a combination of four different instruments. Something that takes a little getting used to is that computers start counting with zero (0). The four voices are numbered 0,1,2, and 3.

The second number is the **note value** that tells the Atari what frequency or tone the note should be. Changing this number is like sliding our twangy ruler off the edge of the table. The larger the number in the SOUND command, the longer the vibrating part of the ruler and the **lower** the note will be. The numbers used by the computer do not match piano musical notes, but we have the right numbers figured out for you in our songs, and we will show you how to translate sheet music to your Atari. The highest note the Atari can make, has the number value zero, and may be too high for most people to hear. There are 256 possible notes, but the highest number (lowest note) we can use is 255 because we started counting at zero. If you use a number larger than 255 the computer will start counting over, and use 256 as zero.

Control Key Magic

While we try out more possibilities of the SOUND command, you can use some very helpful editing features of your Atari. Edit keys will let you try lots of different sounds without typing in the command every time. The secret to editing on the computer is moving the cursor.

Move the cursor by holding down the <CTRL> key and using the arrow keys (located next to the <RETURN> key) to move the cursor around. When the cursor is anywhere on a line and you press <RETURN> the entire line is entered into the computer.


Alle Menschen werden Brüder...
ATARI

Let's try this with our SOUND command. First press the SHIFT and CLEAR keys at the same tine to clear off the screen. Type:

```
SOUND Ø,121,1Ø,Ø8 <RETURN>

END <RETURN>
```

Now hold down the <CTRL> key and press the up arrow until the cursor is on the SOUND command and press <RETURN>. The sound will start. The cursor then jumps down to the line that says END. Press <RETURN> again and the sound will stop just as if you had typed in the commands all over again.

Before you do this section turn the volume on your TV or monitor down low. With four SOUND commands on the screen at once we can try out all the voices and lots of tones without retyping the commands. Press <CTRL> and the up arrow key to move the cursor to the END command. Type the following:

```
SOUND Ø,182,1Ø,Ø8 <RETURN>

SOUND Ø,Ø81,1Ø,Ø8 <RETURN>

SOUND Ø,Ø53,1Ø,Ø8 <RETURN>

END <RETURN>
```

If you make mistakes and get any ERROR messages, use the arrow keys to go to the line with an ERROR and hold down the SHIFT and the DELETE keys together to make the line disappear. It is important that all four SOUND statements are on the screen at the same time with no ERRORS in between. The screen should look like this:

```
READY
SOUND Ø,121,1Ø,Ø8

READY
SOUND Ø,182,1Ø,Ø8

READY
SOUND Ø,Ø81,1Ø,Ø8
```

```
READY
SOUND 0,053,10,08

END
READY
```

Now you can turn the volume back up to normal. With our four SOUND statements on the screen and our <CTRL> key finger all limbered up we can try out dozens of possibilities for the Atari music-making voices. First try turning the sounds on in a different order. Hold down the <CTRL> key and press the up arrow until the cursor is on any one of the sound statements. When you press <RETURN> that sound will play. Now let's try changing some of the numbers carefully without messing up the screen. Move the cursor to the beginning of the second SOUND command and press the right arrow key to put the cursor over the first number. Change the 0 to a 1. Don't press <RETURN> yet. Be sure your finger is on the <CTRL> key when you are pressing the arrow keys and off the <CTRL> key when you are typing the new number. Now move the cursor to the first number in the third SOUND statement and change that to a 2. The changes will not take effect until you type <RETURN>. What we are changing is the voice number. Change the last SOUND statement so the voice number is 3 and the screen should look like this:

```
READY
SOUND 0,182,10,08

READY
SOUND 1,121,10,08

READY
SOUND 2,081,10,08

READY
SOUND 3,051,10,08

READY
END
```

Move the cursor back up to the first SOUND command and press RETURN over and over as it jumps to each new command. Can you hear the difference? All of the "voices" are "singing" together. As you execute the four sound commands in succession the notes join together until all four are playing at once. If the notes

combine into a pleasant sound a musician would call it a chord. Try different note values in the four SOUND commands. Do this by using the editing keys to change the second value in each SOUND command. Notice that the pleasant combinations are hard to find. Note values that are close together are the least pleasant. If you use the same note in two different voices the sound is richer.

The third number of the SOUND command is called **distortion**, because it messes up the pure tone and makes a sound like a sick electric fan. Try the different distortions on one of the SOUND statements. The values 10 and 14 both produce pure notes. The odd numbers make no sound at all, and numbers larger than 14 just repeat the same sounds as 0 to 14. Try a SOUND statement with 08 in the third location.

```
SOUND Ø,121,Ø8,Ø8
```

Distorted sounds are used to make the whoosh of a rocket ship or the sound of a race car in arcade games. Distortion is not very useful to our music program, but we do present a few sound effects in Chapter 8 that use distortion.

The final number in the SOUND command controls the **loudness** or volume. The loudness value does not effect the volume control on the TV speaker, only the loudness of one voice in relation to the others. That is why we need all the voices on the screen to demonstrate how loudness works. Try a loudness of 14 for the first voice and 04 for the other voices. The range of volume is 0 to 15. The total of all the volume settings in the four voices should not be larger than 32. If all four voices are singing together at the same volume they should be set to 8.

Press the SHIFT key and the CLEAR key simultaneously to erase the screen and get ready for the next chapter.

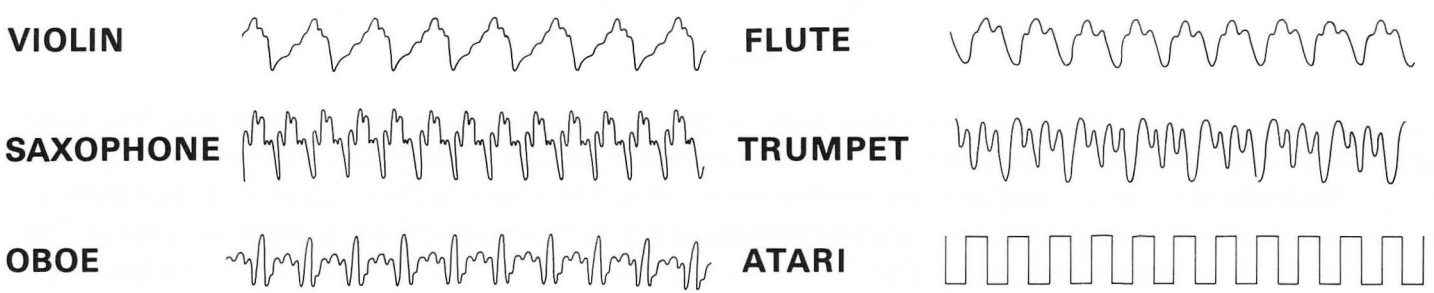
2

RESONANCE AND HARMONY

More Ingredients for Making Music

The musicians of the ancient world discovered that if you have two strings on an instrument tuned to the same note, when you pluck one of the strings, the other string will start to vibrate or "resonate" along with the first. If you have a good bell or a wine glass with clear ring, you can find the right sound on the Atari to make the bell or glass vibrate in sympathy with the Atari sound. The largest part of a musical instrument is the body that resonates with the sound making mechanism. In a trumpet, the mouthpiece makes the sound, but the brass pipes resonate only with certain notes. The sound from the mouthpiece is made louder and purer by the resonating pipes and the stray notes are dampened so they don't come out. The value of a violin is determined by the range and richness of the resonant wooden body that amplifies the sound of the strings.

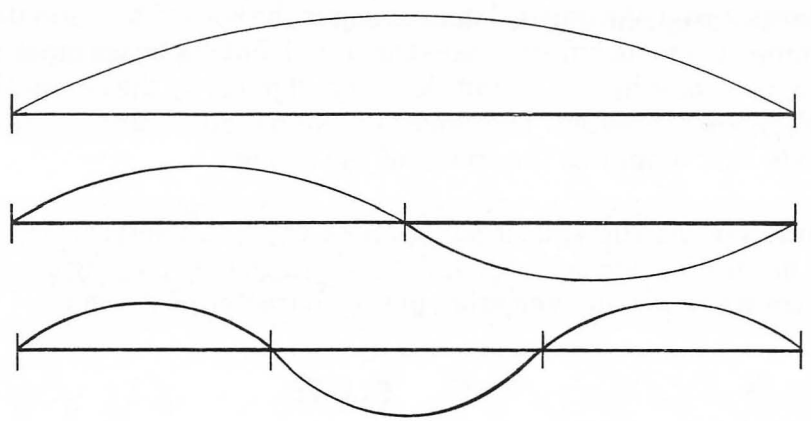
The resonant parts of different instruments create a very complex waveform that is very hard to imitate with synthesizers. Look at the diagram below of different instruments playing the same pitch. Notice that the Atari creates a square wave with no variation in the character of the note.



Some computer musicians try to imitate sounds of the orchestra, but creative musicians realize that the computer can create sounds that never existed before. These new sounds are developing into a new kind of music, which is outside traditional categories.

Harmonic Vibrations

The principles of harmony were first explained by the Greek philosopher Pythagoras in about 550 B.C. He discovered that when one string was twice as long as another, they would vibrate together producing a low note and a high note that sounded very pleasant together. Of course, the musicians of his day already knew this "by ear," but Pythagoras carefully plotted the mathematical basis for string instruments. The lute, which was common during Pythagoras' time, and today's guitars don't have strings with different lengths, but with varying tension. Pythagoras used weights to measure the tension which strings needed to vibrate in harmony. A string would need a weight that was twice as heavy to produce the high note that harmonizes with another string of the same length. A string can also vibrate twice as fast as normal if it is touched lightly, exactly in the center. Each half of the string will vibrate with the next higher harmonic note.



A string can be made to vibrate in two, three, four, or five equal divisions known as overtones. The Atari puts out a pure tone with no overtones, but we still need to know about harmony and overtones because harmony is the basis of our music and our system of musical notes. The notes on the musical scale and the keys on the piano are divided up so that every eight whole notes are an overtone of the note eight keys lower. The black keys on the piano are half steps. The fractional steps of an octave and the equivalent divisions of a vibrating string are shown in the following:

Note	C	D	E	F	G	A	B	C
Frequency	1	9/8	5/4	4/3	3/2	5/3	1 5/8	2
Vibrating Length	1	8/9	4/5	3/4	2/3	3/5	8/15	1/2

You can spend a lifetime learning all the theories of harmony, but this book has only the smallest dose of theory necessary for playing chords. Chords are groups of harmonious notes that play in the background. The melody is the main part of the music. One chord is usually played for several notes of melody.

Harmony on the Atari

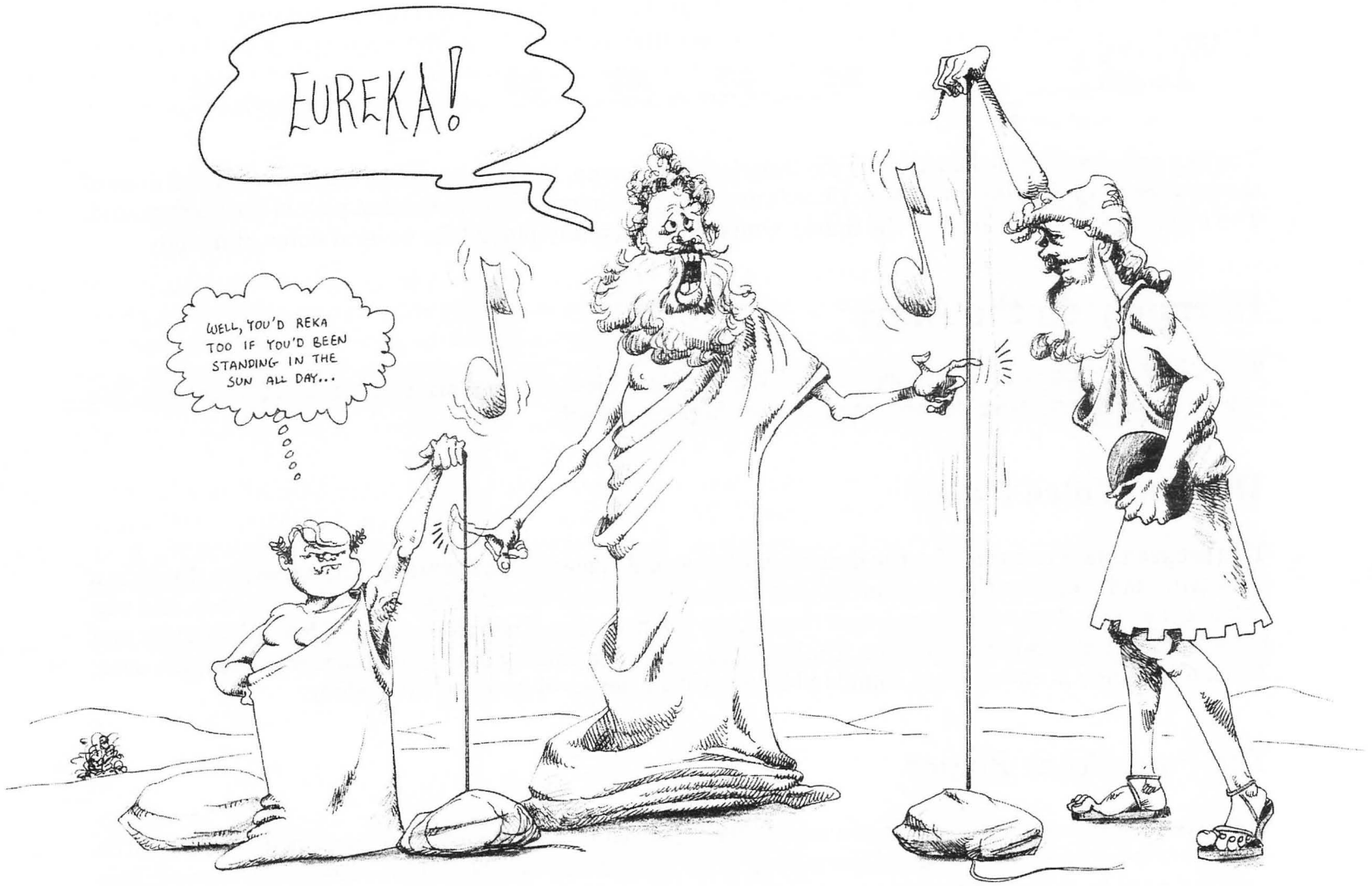
There are four programs in this book that utilize the four voices of the Atari to play music in harmony. You will be writing and using the programs in the chapters ahead.

The Two Voice Player

This program uses one voice for the melody, and a second voice for the harmony. The notes are stored in a data table and read into the program as they are needed. The program is explained step-by-step, and you will learn what a data table is and how it works in the program. The melody notes play individually and there is a short silence between notes. The harmonic notes continue until another harmony note takes over. Thus, a harmony is chosen that sounds pleasant with a series of notes in the melody.

The Four Voice Player

In the four voice player three voices are used for harmony and the three notes are combined in a chord. The chord continues for several notes of the melody until replaced by another chord. Most of the music in this book is played by this program, and most piano and guitar music in other song books will be easily played with the four voice player.


EUREKA!
WELL, YOU'D REKA TOO IF YOU'D BEEN STANDING IN THE SUN ALL DAY...

The Round Player

Familiar rounds such as "Row Row Row Your Boat" rely on the principle of harmonizing the second chorus with first chorus, so that when the first singer gets to "merrily merrily," and the second singer starts "row row," the "row" and "merrily" form a harmonious combination. The round player program in this book starts the melody on voice one, repeats it on voice two, and repeats it again on voice three and voice four. Some fancy computer juggling is used to accomplish these repeats, and a separate player program has been written just for rounds.

The Atari Keyboard Organ

In order for you to play chords in your own music we have written a program for the Atari that plays chords for you. The last music program is the Atari keyboard organ, which transforms the Atari into a chord organ. Six of the most useful chords are programmed onto keys 1-6, and the keys of the alphabet become musical notes. You can try out melodies with different chords. The notes are printed on the screen as you play, so if you play tunes by ear you can easily find the Atari sound values to use in the four voice player and program in the chords you like.

The relationship between the keys on a piano, the notes on sheet music, and the Atari SOUND numbers are on a chart in the section called "From Notes to Data" in Chapter 6. You will be able to use this chart to translate most of the music written for piano, guitar, or voice directly to the Atari. The songs in this book have been arranged especially to take advantage of the four Atari voices for harmony. The Atari sound values and time are included in the music.

3

BASIC PROGRAMMING AND MUSIC

In this chapter you will learn how to program in BASIC in order to keep time so you can make music. There will also be more practice with editing commands that make programming easier.

The ability to beat out a rhythm regularly and accurately seems to be built into human beings. Cultures we describe as "primitive" often have music and dance rhythms that are far richer and more complex than the 3/4 and 4/4 time that is the basis of most Western music. Your Atari also has the ability to keep time. An extremely accurate clock chip that regulates all the computer's activity makes this possible. We have to put SOUND commands into a program in order to utilize this Atari clock.

A program is a series of commands that is organized in a list and stored in the computer's memory. We will be writing our music-playing programs in the computer language called BASIC. We can write a program that will repeat a sound over and over by using a loop to go back to the SOUND command. A loop is any command or arrangement of commands that directs the program to repeat a function or retrace a series of steps. In order to hear a note play repeatedly, we have to turn it on and off. You used the END command to turn off the sound in immediate mode, but this command won't work in a program. The END command will not work as a note stopper because the program ceases completely after END. What we will use is another SOUND command with zeros representing values for the note and loudness. Enter the program below, and you will hear what a loop can do for a program:

```
100 SOUND 0,121,10,10 <RETURN>
120 SOUND 0,0,0,0 <RETURN>
300 GOTO 100 <RETURN>
```

The computer operates so fast that the sound will go on and off like a buzz saw, but we will learn how to fix that soon.

Atari BASIC keeps the program commands in order by line numbers. Notice that in numbering the program lines we skipped a lot of numbers before the first line and between commands. This will allow us to add more commands to the program later if we need to. GOTO is the command that sends the program back to a certain line to start the loop over again. We gave the GOTO line a large number so we could add quite a few lines to the program before it starts over. You will not hear any sound until the program is started. The program will stop when you press the <BREAK> key. We thought you should know this before we told you how to start the program, otherwise you might think your computer was going crazy and pull the plug. The sound might even continue after you press the <BREAK> key, if the computer stops at the SOUND statement. If this happens type END and press <RETURN> to stop the sound. Start the program by typing RUN and pressing <RETURN>.

The Beat Program

In order to hear anything that sounds like a beat, we have to put in a delay that gives the note a chance to play, and another delay to turn off the computer for a rest. This delay can be put in your program with a command known as a FOR/NEXT loop. You can add a line to your program by typing a program line on any clean place on the screen. The computer will put the line in its proper place automatically according to the line number you assigned to the command. This is the FOR/NEXT loop that will make your program slow down between beats:

```
110 FOR PLAY=1 TO 4 : NEXT PLAY <RETURN>
```

The word PLAY is not a computer term: it functions as a variable which can have any name — usually one that helps us remember what the variable does. The Atari does not care what we call our variables, as long as they are not the same as the commands in BASIC. We could have written:

```
110 FOR MOUSE=1 TO 4 : NEXT MOUSE <RETURN>
```

We also need a FOR/NEXT loop, like the following, to make the computer pause between notes.

```
130 FOR WAIT=1 TO 55 : NEXT WAIT <RETURN>
```

You could have used PLAY again or even MOUSE, but WAIT helps us remember what the line is for. Type LIST and press <RETURN> for a fresh listing of the program. It should look like this:

```
100 SOUND 0,121,10,10
110 FOR PLAY=1 TO 4 : NEXT PLAY
120 SOUND 0,0,0,0
130 FOR WAIT=1 TO 55 : NEXT WAIT
300 GOTO 100
```

Notice that the lines are in proper numerical order even though some lines were added later. When you run this program you should hear an even tapping sound like the sound a drummer makes by hitting the rim of his drum. If the program doesn't run correctly, carefully check all the commands and punctuation. There is no way to know in advance what errors you might make.

You can try out different combinations of time and notes by LISTing the program and editing the lines, but there is also an easier way. You can change the notes and times to variables, and assign different values to the variables. Learning to use variables is very important for the programs that follow.

Type <CTRL> and <CLEAR> at the same time. Then type:

```
LIST <RETURN>
```

Use the <CTRL> key and arrow keys to put the cursor on the note value in line 100. Change 121 to N and press the space bar twice to erase the second two digits of 121. Then press <RETURN>. Change the 4 to P in line 110 as follows:

```
110 FOR PLAY=1 TO P : NEXT PLAY <RETURN>
```

In line 130 change 55 to W. Remember to press the space bar to erase the extra digit of 55.

```
130 FOR WAIT=1 TO W : NEXT WAIT
```

Now use a separate line to assign the variables:

```
80 N=121 : P=4 : W=55
```

LIST the program. It should look like this:

```
80 N=121:P=4:W=55
100 SOUND 0,N,10,10
110 FOR PLAY=1 TO P : NEXT PLAY
120 SOUND 0,0,0,0
130 FOR WAIT=1 TO W : NEXT WAIT
300 GOTO 100
```

Once this program runs correctly we'll go on to something that seems like magic. By editing a program *line number* you can make a duplicate of the line. Your program will contain the original line *plus* an exact copy in another part of the program. Move the cursor to the beginning of line 100. Type 140 over 100 and press <RETURN>. The cursor will jump down to the next line. Change 110 to 150 and press <RETURN>. Change 120 to 160, 130 to 170, and 140 to 180. LIST the program. It should look like this:

```
80 N=121:P=4:W=55
100 SOUND 0,N,10,10
110 FOR PLAY=1 TO P : NEXT PLAY
120 SOUND 0,0,0,0
130 FOR WAIT=1 TO W : NEXT WAIT
140 SOUND 0,N,10,10
150 FOR PLAY=1 TO P : NEXT PLAY
160 SOUND 0,0,0,0
170 FOR WAIT=1 TO W : NEXT WAIT
180 SOUND 0,N,10,10
300 GOTO 100
```

We will continue this process until there are four sets of SOUND commands so that we can try combinations of 4/4 time. Move the cursor to line 100 and make the following changes. Remember to press <RETURN> each time.

Change 110 to 190
Change 120 to 200
Change 130 to 210
Change 140 to 220

Change 150 to 230
Change 160 to 240
Change 170 to 250

LIST the program again. In line 140, change N to N1. You do this by moving the cursor over the comma after N and pressing <CTRL> at the same time as the INSERT key. The list below shows the steps you have take to accomplish this. It's easy.

```
140 SOUND 0,N,10,10      original line
140 SOUND 0,N,1010       cursor over comma
140 SOUND 0,N  ,10,10    space added
140 SOUND 0,N1,10,10     N changed to N1
                         Press <RETURN>
```

Change N to N2 in line 180
Change N to N3 in line 220

Add values for your new notes in line 90:

```
90 N1=20:N2=200:N3=160 <RETURN>
```

There is another BASIC command you need to add comments and titles to your program, and that is REM for remark. With this command we can put a title to our program:

```
70 REM *** THE BEAT ***
```

LIST the program. It should look like this:

```
70 REM *** THE BEAT ***
80 N=121:P=4:W=55
90 N1=20:N2=200:N3=160
100 SOUND 0,N,10,10
110 FOR PLAY=1 TO P : NEXT PLAY
120 SOUND 0,0,0,0
130 FOR WAIT=1 TO W : NEXT WAIT
```

```
140 SOUND 0,N1,10,10
150 FOR PLAY=1 TO P : NEXT PLAY
160 SOUND 0,0,0,0
170 FOR WAIT=1 TO W : NEXT WAIT
180 SOUND 0,N2,10,10
190 FOR PLAY=1 TO P : NEXT PLAY
200 SOUND 0,0,0,0
210 FOR WAIT=1 TO W : NEXT WAIT
220 SOUND 0,N3,10,10
230 FOR PLAY=1 TO P : NEXT PLAY
240 SOUND 0,0,0,0
250 FOR WAIT=1 TO W : NEXT WAIT
300 GOTO 100
```

If you were successful, you should hear a little Latin rhythm when you RUN this program. In the following section we talk in-depth about saving programs with a program recorder or disk drive, but if you want to try it now the command is:

SAVE"C:THEBEAT" <RETURN> (for cassette)

SAVE"D:THEBEAT" <RETURN> (for diskette)

Saving Your Music on Disk or Tape

After you spend time working on the Atari player programs you should learn how to save the part that you have done on a cassette tape with the program recorder or on a floppy disk with the disk drive. There are several things that can happen that will make the computer lose what you are presently working on, such as accidentally opening the lid to the computer or unplugging the computer. You can save a program that is only half done, or has only part of a DATA table of notes. It is a very good idea to save what you have done whenever you get up from the computer for any reason. A program that has been saved can be reloaded into the computer and changed or added to at any time. The program in the next chapter should be the first one you save. We will show you how to add to this program to make it play more voices and have more features. It is important to save it once you have done this.


FALLING STAR
TE TUM RUM
TE TUM RUM
TE TUM RUM

We don't want to get the cart before the horse and ask you to save programs that aren't even written yet, but you cannot turn on the disk drive and format a disk after a program is in the computer. Skip to the next chapter if you are already familiar with saving and loading programs, otherwise read the part that applies to your computer system:

Using the Disk Drive

In order to use the disk drive the drive power switch must be turned on before the computer power switch is turned on. There is a little computer board inside the disk drive that has to be functioning before the computer will recognize the drive. It is a good idea to turn on the disk drive power BEFORE inserting your disk. The disk drive motor will hum for a little while, then the "busy" light will go out and the motor will stop. Now put in the disk with Atari DOS (Disk Operating System) that came with your disk drive. The label side should be up and toward you. Never touch the magnetic material that shows in the window of the disk, or the data stored on that part of the disk may be destroyed.

The Atari computer should have the BASIC Computing Language cartridge in the cartridge slot BEFORE the power is turned on. Turn on the power switch on the side of the Atari, and the drive will spin again. In a short while the READY prompt should appear. Type DOS and press <RETURN>. The Disk Operating System will load from the disk. A menu will appear with the DOS utilities. These are programs that help you use the disk system. If you want a catalog of the programs on the disk, type A and press <RETURN> twice. A list of the programs on your disk will appear.

What you need for saving programs is a new disk that has been formatted and has a copy of DOS on it. Select "I" from the DOS menu and follow the directions on the screen to format a new disk. After the new disk is formatted, copy the DOS files onto this new disk by selecting "H" from the DOS menu and following those directions.

To run one of programs, type B <RETURN> to go from DOS to BASIC. You will know you are in BASIC if you see the word READY and a small white square cursor.

Type:

```
LOAD"D:NAME" <RETURN>
```

Where we have NAME you should type in the name of the program you want to load, exactly as it appears on the disk catalog. When the READY prompt comes back you can type RUN and press <RETURN> or LIST and <RETURN>. When you have finished writing or editing your program or the DATA for your music you can save the program by typing:

```
SAVE"D:NAME" <RETURN>
```

If you use a name you have already used the program you save will replace the original program that had the same name. The original will be gone forever. You can also type:

```
SAVE"D:NEWNAME" <RETURN>
```

This will leave the program called NAME unchanged on the disk and create a new program called NEWNAME on the disk along with NAME.

Using the Program Recorder

You can save the program currently in the computer's memory on a cassette by typing:

```
SAVE "C:NAME" <RETURN>
```

Where we have NAME you can have any eight letters or numbers. The name must start with a letter and not contain spaces or punctuation. When you press <RETURN> the computer will beep its internal speaker twice as a signal for you to get a blank tape ready to record. Start the tape recorder and press <RETURN> to start recording the program. You will be able to hear the data being sent. If you want to save several programs on one tape make note of the tape counter numbers and write this on the cassette label.

To put the program back into the computer, type NEW, press <RETURN>, and make sure you have a READY prompt on the screen.

Type:

```
LOAD"C:NAME" <RETURN>
```

Make sure the tape is at the right location for the program you want. After the computer beeps, press play on the program recorder and press <RETURN> again. You will be able to hear the program loading. When the sound stops the tape will stop as well. Press the STOP lever on the recorder. You can RUN the program or add more to it if you wish.

4

STARTING YOUR OWN PLAYER PROGRAMS

A Little History

In this chapter we'll get a good start on the programs that will be used to play the songs in this book on the Atari. In the process you will get a little background in computing and enough about BASIC to understand the programs you will be writing.

The role that music played in the history of computer technology is a fascinating subject. Music is a subtle blend of intuition and science. While it is true that the player piano and the Atari lack the warmth and passion of a human musician, there is a long history of music written for mechanical instuments. If you have ever seen the inside of a player piano and looked at the piano roll, you have probably noticed that the roll is similar to a computer punch card. Both player pianos and punch card readers trace their origins to the French inventor Joseph Marie Jacquard (1752-1834). By 1802 he perfected a loom that stored data on punched cards for weaving intricate designs in many colors. Jacquard was improving on a loom built by a great maker of automata named Vaucanson, who invented automatic looms as early as 1745.

This was a period of great inventiveness in Europe. Fantastic music boxes and animated dolls called automata were the wonders of their time. Mozart, who lived at the same time as Jacquard, composed music especially for the finest of these music boxes, which were really entire robot orchestras complete with musician dolls that played tiny instruments.

Jacquard's punched cards were adapted by dozens of inventors of musical instruments. A patent from 1865 in the name J. Amman describes the use of punch cards to control the contacts of an electromechanical

player piano. At the turn of the twentieth century an enterprising inventor named Herman Hollerith used the punched card and solenoid system to tabulate statistics for the United States Census, and out of this the IBM company was born. The player piano mechanism is so similar to early computer memory that piano rolls were actually used to run special IBM electric typewriters called Robotypers as late as the 1960's. Punched cards or punched tape was the chief means of storing information for the computer until very recently, and is still used when accuracy and permanence are critical. The information that is stored on these cards is called "data."

Just as the punched card could be used to make a piano play, we will be using computer data to play music on the Atari. The floppy disk or cassette tape has replaced the punch card as a way of storing computer data, and that makes it even easier to use data to play music. The actual SOUND values and the values for musical time will be stored at the end of the program in lines of data called a "data table." The player piano required a special mechanism to "write" the piano rolls and the piano itself was "dedicated" to the task of playing these rolls. A great advance made in computers in the 1950's was combining the data with the instructions telling the computer what to do with this data. Now we can not only write the music for the Atari, but we will also write the "piano" to play it on. What we will learn to do in this chapter is write a program that can read notes from a DATA table and then play music.

The player piano program that you will write will be expanded with a few additional commands to play two voice harmony and four voice harmony. You can save the player program without notes and use it for all the songs. After the notes have been entered into the DATA table the program is saved again under the name of the song. Each song will have its own player attached.

BASIC programs are organized by line numbers and run in numerical order. You can start writing a program with any line number and add lines in any order. When you LIST or RUN the program the computer automatically puts the program in the correct order starting with the lowest line number. Because the computer sorts the lines for us and allows us to skip numbers, we can start with a few "bare bones" of the program and add features as we go along. This will allow you to RUN the program each time you add a new command and make sure each part works before continuing. You can SAVE the one voice player on tape or disk and then use the program as a basis for the two voice program. The two voice program can then be saved and a few lines can be added to make the four voice program. By building more complex programs out of simpler ones you save lots of time typing and reduce the possibility of error.

Look at the four voice program at the end of the Chapter 6 to see what our ultimate goal is. If you started by typing this program from beginning to end you could not RUN it until you reached line 630, and then if there were a typing error, you might not know where to look to fix it.

If you are using a disk drive be sure the drive is on and you have a formatted blank disk in the drive.

The One Voice Player

In "The Beat" program, you saw how a program keeps time by looping through the program and waiting by means of a FOR/NEXT loop. It would be impractical to keep adding SOUND statements for every note in a melody. Several features of BASIC give us powerful tools to play the notes more efficiently. The core of all of our player programs is a SOUND statement with a variable for the note value. Let's start with this statement. First type:

```
NEW <RETURN>
<CTRL> <CLEAR>
480 SOUND 0,NT,10,10
520 REM PLAY ONE VOICE
```

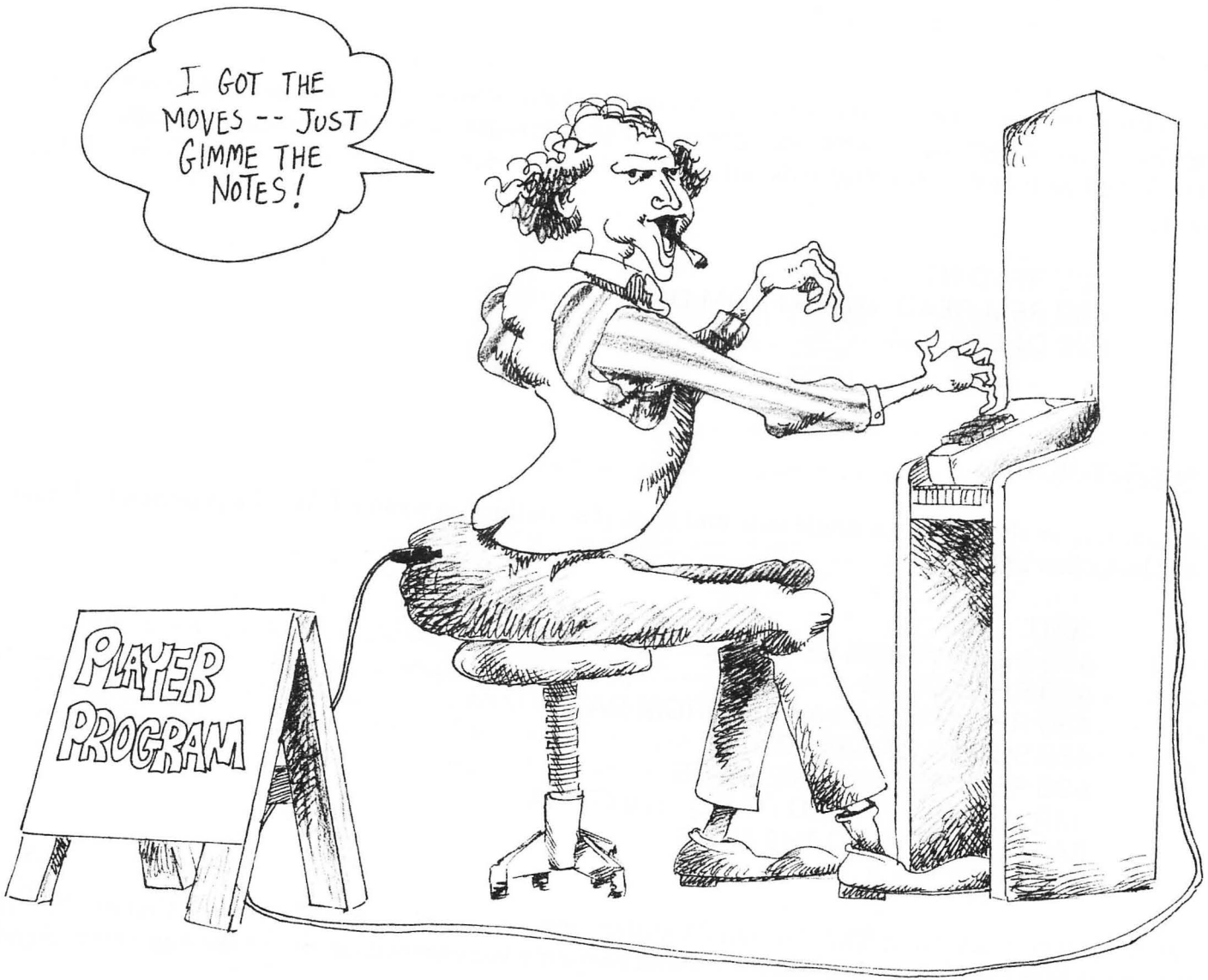
Now add a delay loop to hold the note long enough to hear:

```
530 FOR PLAY=1 TO (TM * L):NEXT PLAY
540 REM HOLD THE NOTE
```

The playing time will be regulated by two variables, L and TM, that are multiplied together. L will be set in the beginning of the program and regulates the time of the entire melody, like a metronome. One L will be the shortest interval that a note plays, and will be equivalent to an eighth note. Each note will be accompanied by an equivalent time value TM. If L is set to 50, and a note has a TM (time) value of two the computer will wait for 50 times 2 or 100 counts. L is set in line 400.

```
400 L=50
410 REM *** SETS LENGTH OF NOTES
```

If you RUN this short three line program you will not hear any sound because NT and TM are set to 0 by the RUN command. You can test even this small part of the player program by giving values to NT and TM in immediate mode.


I GOT THE MOVES -- JUST GIMME THE NOTES!
PLAYER PROGRAM

```
(no line number) TM = 2 <RETURN>
(no line number) NT = 121 <RETURN>
(no line number) GOTO 400 <RETURN>
```

You have to use GOTO instead of RUN when you set variables directly because RUN clears out variables before a program is run. Try using the <CRTL> key and the up arrow key to change TM and NT to something else. Press <RETURN> with the cursor next to GOTO 400 to replay the program with the new values. With just a few more commands our player will be retrieving the notes and time from a DATA table. Type:

```
420 READ NT,TM
430 REM READ NOTES FROM DATA
800 DATA 121,4
```

Type:

```
RUN <RETURN>
```

Now your player should play a single note and stop. If something is wrong, LIST the program and make sure it looks like this:

```
400 L = 50
410 REM *** SETS LENGTH OF NOTE
420 READ NT,TM
430 REM *** READ NOTES FROM DATA
480 SOUND 0,NT,10,10
520 REM *** PLAY ONE VOICE
530 FOR PLAY = 1 TO (TM*L) : NEXT PLAY
540 REM *** HOLD THE NOTE
800 DATA 121,4
```

A common error when using READ or DATA statements is placing a comma before the first or after the last element that these statements use. A comma goes only between values, not at the beginning or end.

Now we can add more notes and put in a GOTO loop to get a new note each time around.

```
630 GOTO 400
800 DATA 96,1,108,1,121,1,108,1
```

If you RUN this partial program you will play as much music as the DATA statements contain and then the program will "crash," as programmers say. The screen will display:

```
ERROR-- 6 AT LINE 420
```

This is the OUT OF DATA error that occurs when the READ statement reads past the end of the DATA. The program does not END properly so the last note continues to play. We will put in a routine to end the program and build in a way of repeating the music if you desire, since many songs repeat. The program looks for a "flag" that indicates the end of the DATA. In our player a note of value 1 is never used, so we can use it as a flag. Pick a line number that is way above the number of lines of DATA you will need and put in the "LAST DATA."

```
1100 DATA 1,0
```

Right after line 420 reads the DATA, line 440 will look for the number 1. If it is found the DATA will be RESTOREd which means that the READ statement will start at the beginning of the DATA again. Line 440 also counts the number of times the DATA is restored by adding 1 to the variable T.

```
440 IF NT = 1 THEN T = T+1 : RESTORE
450 REM *** COUNT NUMBER OF PLAYS
```

The player program at this point will play the same few notes continuously until you press the <BREAK> key.

```
460 IF T = 2 THEN END
470 REM SET NUMBER OF REPEATS
```

Now we have a player program that is beginning to work fairly well. It is important to see how each part of the program contributes to the overall working of the player so that you can confidently change the program to meet your own needs and play more difficult music.

What we need at this point is a printout of the notes on the screen, so that we can proofread the music as we write it.

```
600 PRINT "NT =";NT;" TM =";TM
```

This is a combined print statement that uses a semicolon to keep the elements together on the same line. Don't forget the space before "TM". It makes the printout readable.

RUN the program now and you will get a running report of the notes as they are played. If you hear a sour note, press the <BREAK> key as quickly as possible. The incorrect note should be at the bottom of the list. With a printout for checking errors you can easily write data for more musical notes. The data can be verified at any time by listening to the music and checking the list of notes on the screen against the list below:

```
800 DATA 96,1,108,1,121,1,108,1
810 DATA 96,1,96,1,96,2
820 DATA 108,1,108,1,108,2
830 DATA 96,1,81,1,81,2
840 DATA 96,1,108,1,121,1,108,1
850 DATA 108,1,96,1,96,1,96,1
860 DATA 108,1,108,1,96,1,108,1
870 DATA 121,4
1100 DATA 1,0
```

There is still something which is necessary before we can listen to a correct version of "Mary Had a Little Lamb." Notes that follow one another run together as one long note. To correct this problem we have to add something to the program to shut off notes.

```
700 SOUND 0,0,0,0
710 REM *** TURN OFF MELODY NOTE
```

Then we have a very short wait between notes:

```
720 FOR WAIT=1 TO 8:NEXT WAIT
730 REM *** PAUSE BETWEEN NOTES
```

You can experiment with different values for this WAIT, and put in the value that sounds right to you. When you are writing the songs from the music in this book or from other sheet music you can change any of

the variables "by ear" and have a customized player to go with each song. Add a title to your player program and it will be done.

```
100 REM *** ONE VOICE PLAYER ***
```

After removing all the necessary REM statements, the finished program should match this listing:

```
100 REM *** ONE VOICE PLAYER ***
400 L=50
420 READ NT,TM
440 IF NT=1 THEN T=T+1:RESTORE
460 IF T=1 THEN END
480 SOUND 0,NT,10,10
530 FOR PLAY=1 TO (TM*L):NEXT PLAY
600 PRINT "NT =";NT;" TM=";TM
630 GOTO 400
700 SOUND 0,0,0,0
720 FOR WAIT=1 TO 8:NEXT WAIT
800 DATA 96,1,108,1,121,1,108,1
810 DATA 96,1,96,1,96,2
820 DATA 108,1,108,1,108,2
830 DATA 96,1,81,1,81,2
840 DATA 96,1,108,1,121,1,108,1
850 DATA 108,1,96,1,96,1,96,1
860 DATA 108,1,108,1,96,1,108,1
870 DATA 121,4
1100 DATA 1,0
```

Don't turn off your Atari yet! Make sure you have a copy of this program saved to disk or tape. In just a short while you will turn this one voice program into a versatile instrument for playing the songs in this book.

5

THE TWO VOICE PLAYER

ave the one voice player program on disk or tape with the name ONEVOICE. With a few more lines of BASIC the one voice player will become a two voice player for the first group of songs in this book. To make writing music easier you will have the "players" saved on disk without notes. For each song you will add notes to the appropriate player. Compare the following program to the program on your Atari screen and carefully check each line number. Make the necessary changes for the extra voice.

```
100 REM
110 REM  ****************************************
120 REM *                                        *
130 REM *          The Atari Music Maker         *
140 REM *                                        *
150 REM *             For Two Voices             *
160 REM *                                        *
170 REM *           By Hal Glicksman             *
180 REM *                                        *
190 REM *          ©1983 DATAMOST Inc.           *
200 REM *                                        *
210 REM  ****************************************
220 REM
230 REM *** PROGRAM VARIABLES
240 NT=0:REM NOTE FOR MELODY
250 TM=0:REM LENGTH OF MELODY NOTE
260 A=0:REM NOTE FOR VOICE ONE
290 T=0:REM NUMBER OF REPEATS
```

```
300 REM *** SET VARIABLES TO ZERO
310 PRINT "↑":REM CLEAR SCREEN COMMAND
320 POSITION 2,8
330 REM POKE 752,1
340 REM *** INVISIBLE CURSOR, REMOVE 'REM' FROM LINE ABOVE WHEN
    EDITING IS DONE
350 PRINT "                    "
360 PRINT "                    "
370 PRINT "                    "
380 PRINT "                    "
390 PRINT "                    "
400 L=50:REM SLOW
410 REM *** SETS LENGTH OF NOTE
420 READ NT,TM,A
430 REM *** READ NOTES FROM DATA
440 IF NT=1 THEN T=T+1:RESTORE
450 REM *** COUNT NUMBER OF PLAYS
460 IF T=2 THEN END
470 REM *** SET NUMBER OF REPEATS
480 SOUND 0,NT,10,10
490 SOUND 1,A,10,8
520 REM *** PLAY TWO VOICES
530 FOR PLAY=1 TO (TM*L):NEXT PLAY
540 REM *** HOLD THE NOTE
550 PRINT "NT=";NT;" TM=";TM;
560 PRINT " A=";A
570 REM *** PRINT NOTES ON SCREEN
580 REM INSERT 'REM' IN PRINT STATEMENTS WHEN NOTES ARE CORRECT.
590 SOUND 0,0,0,0
600 REM *** TURN OFF MELODY NOTE
610 FOR WAIT=1 TO 8:NEXT WAIT
620 REM *** PAUSE BETWEEN NOTES
630 GOTO 420
640 REM *** RETURN TO GET NEXT NOTE
650 REM
```

NAME OF SONG

```
660 REM LAST DATA 1,0,0
670 REM
680 REM  ****************************************
690 REM *      DATA TABLE TO HOLD NOTES          *
700 REM *             NAME OF SONG               *
710 REM *                                        *
720 REM  ****************************************
```

Lines 100-220 are the title panel for the program.

Lines 230 to 300 are the program variables. It is good programming practice to set all variables to zero and use REM statements to tell what each variable represents.

Line 330 is a POKE command that makes the cursor invisible when the title screen is on. We start out with this command disconnected by a REM statement because you could not edit notes or program lines with an invisible cursor. If you remove the REM statement and have an invisible cursor you can make it reappear by pressing the <BREAK> key.

Line 420 must be changed as follows to read a note, a time value and a harmony note. Use the edit commands to change the present line, or type a new line 420. Either way the new line 420 will replace the old one in the program.

```
420 READ NT,TN,A
```

Add a SOUND command for the second voice:

```
490 SOUND 1,A,10,8
```

The loudness of the harmony is slightly lower (8) than the melody (10). You are welcome to change this value to suit your taste.

Add a semicolon to the end of line 550.

Line 560 adds a print statement for the harmony notes:

```
560 PRINT " A=";A
```

All that is left to do is add REM statements for the data table in lines 650 to 720.

REM statements are not required to make a program run correctly. But, you will appreciate them because they make it easy to find your place in the program when you want to change a value.

Making the Title Screen

The title of the song is printed on the screen using ATARI control characters, which make a box outline for the title. The screen is cleared in line 31Ø. First write 310 PRINT" then press the <ESCAPE> key once. This tells the Atari to place the next <CTRL> character into the program listing rather than executing the control command. Next type <CTRL><CLEAR>. A bent-back arrow should appear in the PRINT statement on your screen (it will print as a right curly brace on a printer). Close the PRINT statement with a " (quotation mark). You can combine commands on one line if they are separated by a colon. We put a colon and a REM after the clear screen to identify the line. This is what you will see on the screen:

```
31Ø PRINT"↰":REM CLEAR SCREEN COMMAND
```

The next command POSITIONs the cursor on the eighth line from the top and two spaces from the left.

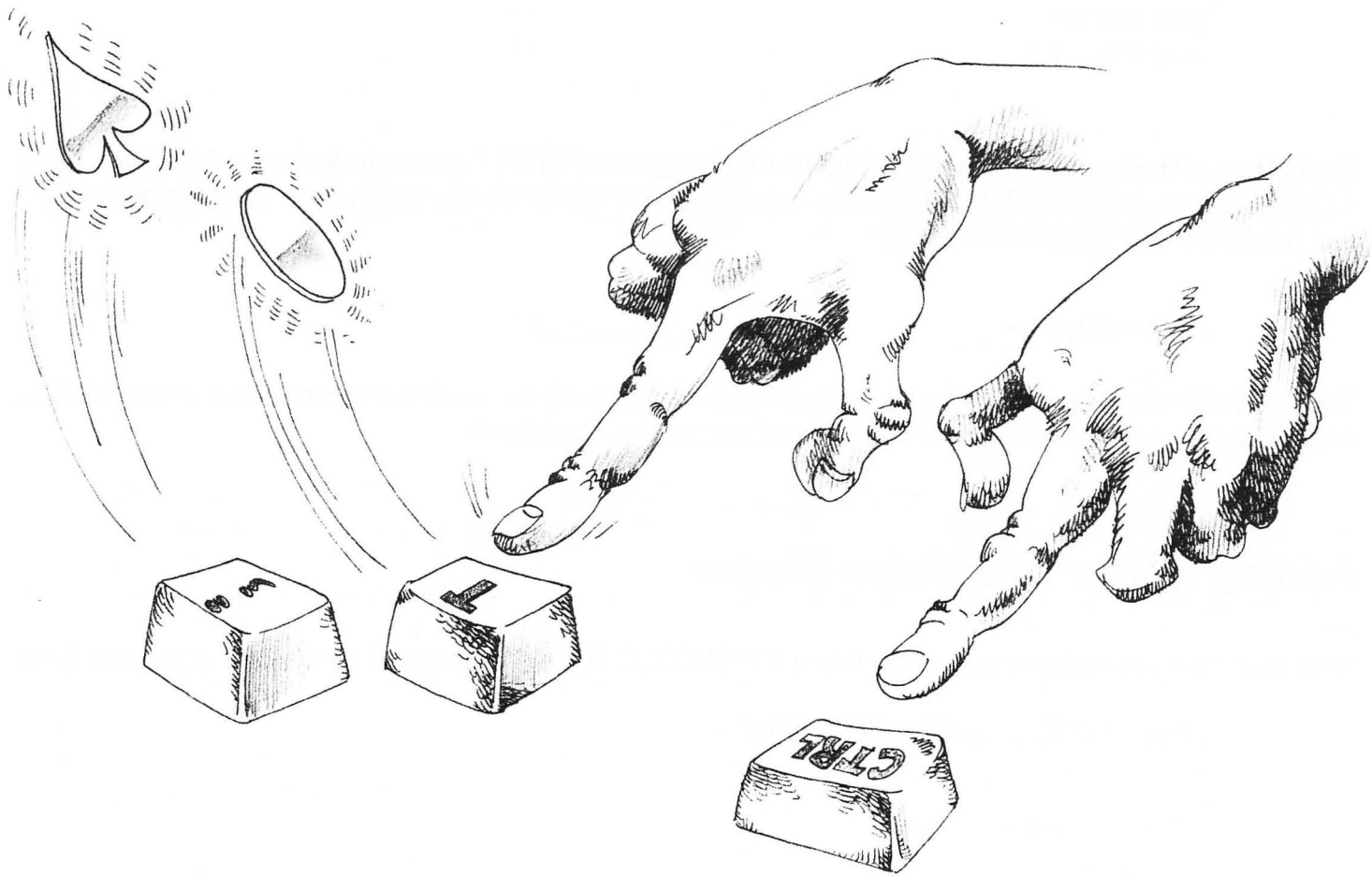
The Atari has a set of graphic symbols that can be used along with the letters and numbers on the keyboard to create charts, title screens and other useful images. These symbols are not printed on the keys, but are displayed in a chart in your *Atari BASIC Reference Manual*. In our music program we will only draw a simple rectangular box around the title of the song, but you are invited to expand on the simple graphics if you wish.

```
35Ø PRINT "                ┌──────────────┐ "
```

To create line 350 type: PRINT " press the space bar ten times, press <CTRL>Q once, press <CTRL>R sixteen times, press <CTRL>E, type the closing quotation mark (") and finally <RETURN>.

```
36Ø PRINT "                |              | "
```

To create line 360 type PRINT", press the space bar ten times, press <SHIFT> [=], press the space bar sixteen times, press <SHIFT> [=] once, and finally press <RETURN>.


CTRL

To make the rest of the box just renumber line 360 as 370 and 380. Use the <CTRL> up arrow to position the cursor at the beginning of 360 and renumber it 370 and press <RETURN>. Renumber 370 again as 380. LIST the program and you will see that lines 360, 370, and 380 are identical except for the line number.

```
360 PRINT "                |                |"
370 PRINT "                |                |"
380 PRINT "                |                |"
```

The bottom of the box is created by typing the following: 390 PRINT ", press the space bar ten times, press <CTRL>Z once, press <CTRL>R sixteen times, press <CTRL>C once, type a closing quotation mark (") and finally press <RETURN>.

```
390 PRINT "                └────────────────┘"
```

You can LIST the program at any time. Use the edit commands to position the cursor inside the title box, and write the name of the song that you want printed on the screen.

Adding to the Two Voice Player

Save your two voice player under the name TWOVOICE. When you want to write a song you can type:

```
LOAD"D:TWOVOICE" <RETURN>

LIST <RETURN>
```

You will have the player in the computer's memory and the program listed up to the line where you start to enter notes in DATA statements. Skip about 100 line numbers between the player program and the start of the data table so you can add to the program later if you want to.

```
690 REM *      DATA TABLE TO HOLD NOTES      *
700 REM *            NAME OF SONG            *
710 REM *                                    *
720 REM ************************************
```

Add your DATA at 800 or even higher.

```
800 DATA 121,4,0,0,0 (and so on)
```

See "From Notes to Data" in Chapter 6 on entering musical notes from sheet music. Number the DATA statements by 10's and end the DATA statement with the last note in a measure. You can make the DATA statements as short as you like to match each measure in the sheet music, and make it easy to keep track of where you are.

Don't try to put more than 2½ computer screen lines of DATA in one statement. The computer will beep at you when you are near the limit.

Before you try to play part of the song, put in a final DATA statement like the one below, which turns off the sound when the notes have been used up:

```
2000 DATA 1,0,0
```

Use a large enough line number for the last DATA statement to allow plenty of line numbers between the player and this last DATA statement.

When the song DATA is entered, or even part of it is done type RUN and press <RETURN> to hear the music. You can save all or part of the program by typing:

```
SAVE"D:SONGNAME"
```

The computer will save the player program and combine it with the notes. Each song that you save will have its own copy of the player attached. When the notes are entered, save the song before you edit the title screen or other parts of the player. This will be a "backup" in case you mess up the player program during editing. Type:

```
LIST 0,400 <RETURN>
```

The above command will LIST the first part of the player with the title screen. Use the <CTRL> key and the arrow keys to put the song name in the title screen box. If the titles are not centered you can insert or delete spaces before or after the title as necessary.

When you are sure you won't be changing the the notes, you can eliminate the PRINT statements that showed the notes on the screen as they were played. Type:

```
LIST 550,560 <RETURN>
```

Move the cursor in front of the PRINT statement in line 510 and type <CTRL> <INSERT> three times. Type REM <RETURN> in front of the PRINT statement. Place a REM in front of the PRINT in line 520 as well. The edited lines should look like this:

```
550 REM PRINT "NT=";NT;" TM=";TM;
560 REM PRINT " A=";A
```

The REM command will disconnect the PRINT statements that displayed the notes while you were writing the music. Without the PRINT statements the music has more accurate rhythm and the title screen stays clean while the music plays.

Line 400 is the variable that controls the overall pitch that the music plays.

```
400 L=50
```

You can type LIST 400 <RETURN>, and change the value of L to make the music play faster or slower. The larger the value of L the slower the music plays.

After your music is saved in final form, you do not have to use LOAD and RUN separately. For example, if you finished writing "America the Beautiful" and saved it with the file name AMERICA, you can simply type:

```
RUN"D:AMERICA" <RETURN>
```

"America the Beautiful" will load and run by itself.

6

THE FOUR VOICE PLAYER

nly a few differences exist between the four voice player and the two voice player you just finished, and you may have noticed that we have been skipping line numbers to allow room for these additional commands. We'll take a few moments now to write the four voice player and then we can go ahead with the music very soon. The four voice player program is shown below and the differences between it and the two voice player are listed following the program.

```
100 REM
110 REM  ****************************************
120 REM *                                        *
130 REM *          The Atari Music Maker         *
140 REM *                                        *
150 REM *             For Four Voices            *
160 REM *                                        *
170 REM *             By Hal Glicksman           *
180 REM *                                        *
190 REM *          © 1983 DATAMOST Inc.          *
200 REM *                                        *
210 REM  ****************************************
220 REM
230 REM *** PROGRAM VARIABLES
240 NT=0:REM NOTE FOR MELODY
250 TM=0:REM LENGTH OF MELODY NOTE
260 A=0:REM NOTE FOR VOICE ONE
```

```
270 B=0:REM NOTE FOR VOICE TWO
280 C=0:REM NOTE FOR VOICE THREE
290 T=0:REM NUMBER OF REPEATS
300 REM *** SET VARIABLES TO ZERO
310 PRINT "↶":REM CLEAR SCREEN COMMAND
320 POSITION 2,8
330 REM POKE 752,1
340 REM *** INVISIBLE CURSOR, REMOVE 'REM' FROM LINE ABOVE
    WHEN EDITING IS DONE
350 PRINT "                    "
360 PRINT "                    "
370 PRINT "                    "
380 PRINT "                    "
390 PRINT "                    "
400 L=50:REM SLOW
410 REM *** SETS LENGTH OF NOTE
420 READ NT,TM,A,B,C
430 REM *** READ NOTES FROM DATA
440 IF NT=1 THEN T=T+1:RESTORE
450 REM *** COUNT NUMBER OF PLAYS
460 IF T=2 THEN END
470 REM *** SET NUMBER OF REPEATS
480 SOUND 0,NT,10,10
490 SOUND 1,A,10,4
500 SOUND 2,B,10,4
510 SOUND 3,C,10,4
520 REM *** PLAY FOUR VOICES
530 FOR PLAY=1 TO (TM*L):NEXT PLAY
540 REM *** HOLD THE NOTE
550 PRINT "NT=";NT;" TM=";TM;
560 PRINT " A=";A;" B=";B;" C=";C
570 REM *** PRINT NOTES ON SCREEN
580 REM INSERT 'REM' IN PRINT STATEMENTS WHEN NOTES ARE CORRECT.
590 SOUND 0,0,0,0
600 REM *** TURN OFF MELODY NOTE
```

NAME OF SONG

```
610 FOR WAIT=1 TO 8:NEXT WAIT
620 REM *** PAUSE BETWEEN NOTES
630 GOTO 420
640 REM *** RETURN TO GET NEXT NOTE
650 REM
660 REM LAST DATA 1,0,0,0,0
670 REM
680 REM  ****************************************
690 REM *      DATA TABLE TO HOLD NOTES          *
700 REM *              NAME OF SONG              *
710 REM *                                        *
720 REM  ****************************************
```

Additions and changes:

```
420 READ NT,TM,A,B,C

500 SOUND 2,B,10,8
510 SOUND 3,C,10,8

560 PRINT " A="A;" B=";" C=";C
660 REM LAST DATA 1,0,0,0,0
```

Though the four voice program is only slightly different from the two voice player the DATA statements will be much longer.

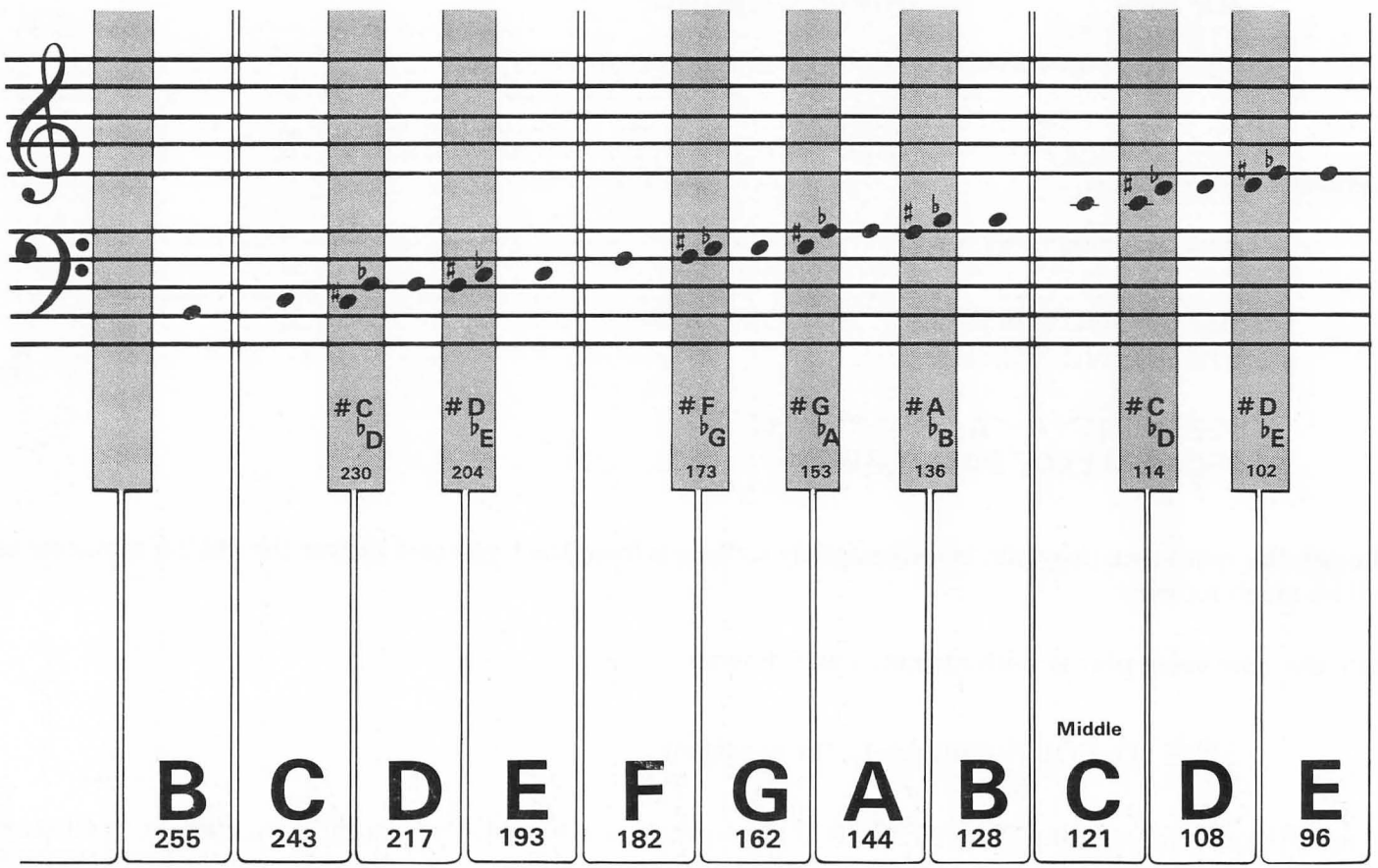
Save the four voice player without notes as follows:

```
SAVE "D: FOURVOIC (or C: for cassette)
```

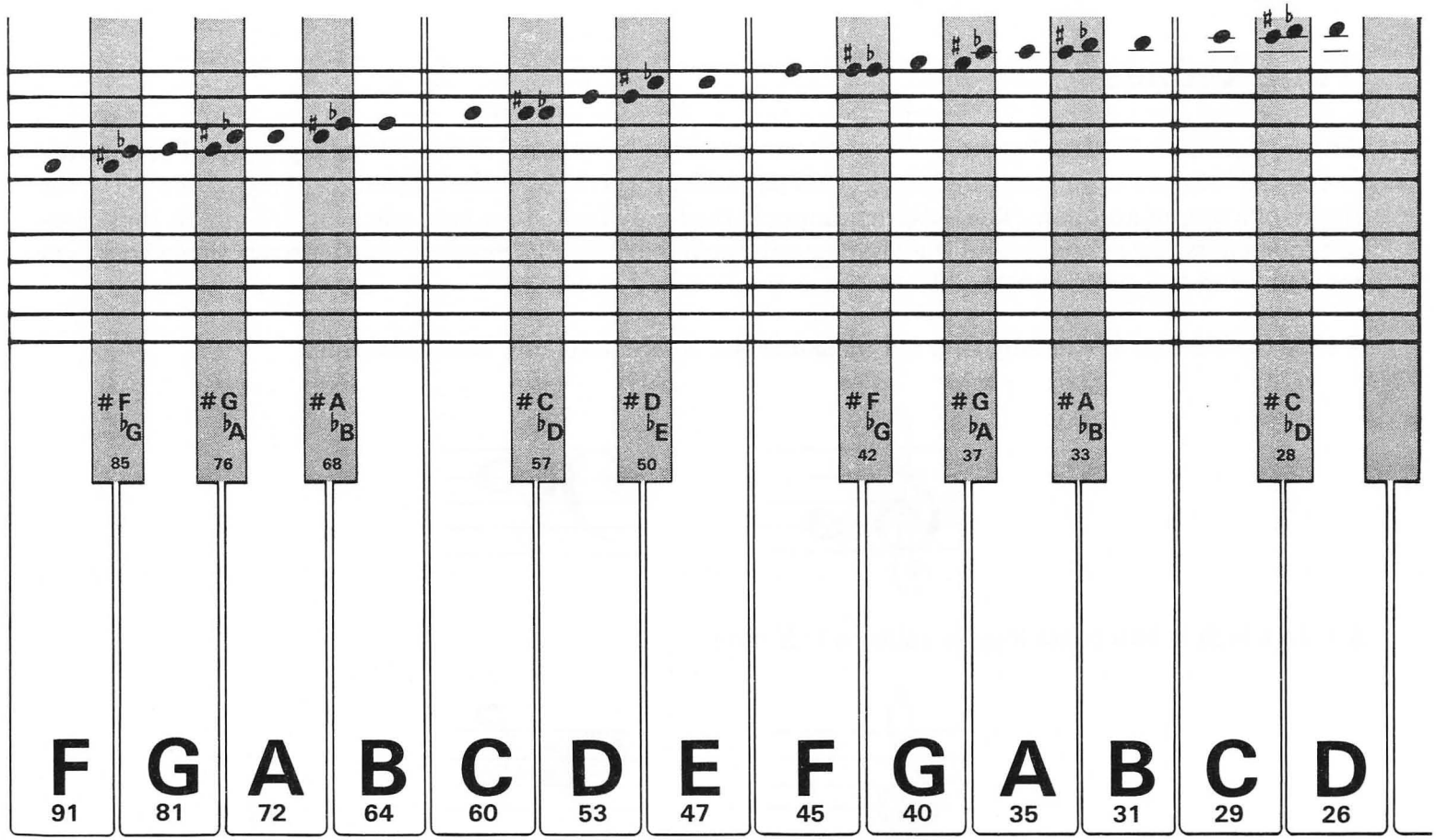
Forgive the strange spelling of FOURVOIC, but program names are limited to eight characters, must start with a letter, and cannot contain spaces, punctuation, or special characters. You can use "D:FOURVOICE but the computer will omit the final E on the diskette catalog.

From Notes to Data

The songs in this book have been specially arranged to take advantage of one voice for the melody and either a one part harmony or three part harmony. The notes have been translated into their equivalent value for the Atari SOUND statement and a time value that works in the player programs. For additional piano, vocal, or guitar music that you may wish to play on the Atari you can translate the music using the charts and instructions in this chapter.



The notes above middle C appear on the upper set of lines (called the treble clef), and notes below middle C appear on the lower set of lines (called the bass clef). The melody is usually played on the treble clef and the harmony on the bass clef, which is the pattern we will use in this book. As you can see from the songs in this book the notes climb above and below the five lines of the musical staff. Some bass notes for the harmony can be higher than than some of the low notes in the melody, so the division at middle C is not an absolute rule.



A # (sharp) sign before a note means that the note to be played is ½ step higher than the natural note. On the piano this means play the next key to the right, whether it is black or white. The ♭ (flat) sign means that the note to be played is ½ step lower than the natural note.

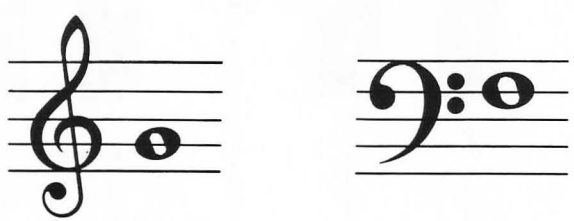
When sharps or flats appear after the clef sign at the beginning of each music staff, the note corresponding to each sharp or flat sign is sharped or flatted for the entire song.

Only if a ♮ (natural) sign appears in front of the normally sharped or flatted note is it played in its natural state, and it is repeated that way only for the duration of that particular measure.

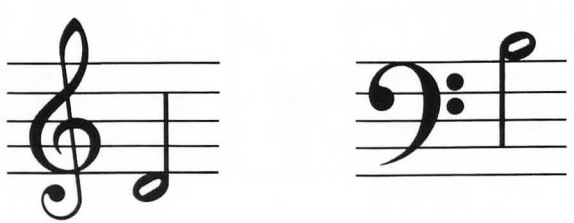
Time Values

The vertical lines separating the notes in the music are the "measure" or equal time segments. The measure is divided into beats, and the number of beats per measure is listed at the beginning of the music as the top figure of a pair of numbers. The bottom number is the kind of note that gets one beat. 4/4 means that there are four beats per measure and the quarter note gets one beat. 3/4 means that there are three beats per measure and the quarter note gets one beat.

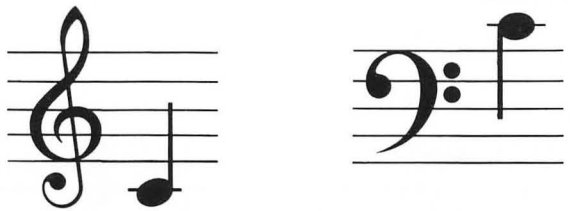
A note that is four beats long in a 4/4 measure is a whole note, and looks like this:



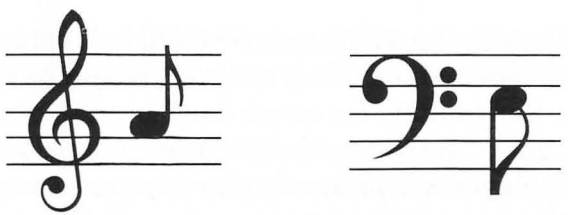
A note which is two beats long is called a half note.



A note which is one beat long is a quarter note.

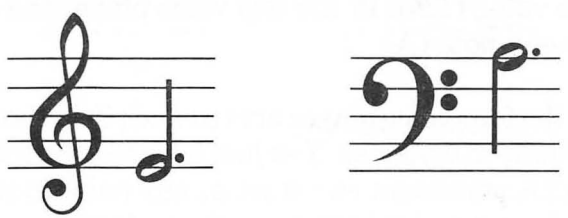


A note which is one half a beat long is called an eighth note.



The number of beats these notes represent can change if the time changes, but the relationship of these notes to each other does not change.

A dot after the note adds one half the time value to the note, so in 4/4 time a dotted half note would be held for three beats instead of two.



Sometimes a song will not start on the beat, so the first measure of the song may not have the full number of beats. If this is the case, the last measure of the song will have the remaining beats.

For some beats, the music may have no tones in the melody. This bit of quiet time is called a "rest," and is measured in beats just like the notes.

whole rest half rest quarter rest eighth rest

As you can see, it is hard to distinguish the whole note rest and half note rest from one another. In 4/4 time the half note rest is held for two beats and the whole note rest would be held for the length of the measure. Don't worry if this is confusing. Fortunately for us we don't need to practice reading music until we can sight read quickly. You will be translating the music into Atari notes slowly and the computer will play the music at any desired tempo. The player will patiently play the music over again until you have gotten the sound just the way you want it. You can edit the data at any time to fix a sour note, a skipped beat, or make a creative change.

Atari DATA Statements

The READ command reads the variables that you need for a each note in the melody. In the one voice player it reads the note (NT) and time value (TM). In the two voice player the READ statement reads the note (NT), time (TM), and one harmony note (A).

Additional notes of harmony in the four voice player are (B) and (C) in the READ statement for that player. The DATA statements have no built-in divisions. You just have to keep track of the notes and time values in a continuous string. A new DATA statement can start at any point in the string of data, and the READ statement will keep on reading data. Atari BASIC has a limit of 128 characters in any one line that applies to DATA statements as well as every BASIC statement. The DATA statements can be any length up to 128 characters, but for your own sanity and ease of correcting it would be a good idea to match each DATA statement with a measure of music. There is no problem having only a few numbers in a DATA statement. The shorter the DATA statement, the easier it will be to find your place in the music.

The Time Value

Fractional notes would make the time value numbers very confusing. An eighth note would have to be written .125 and a 16th note would be .0625. We have written the player program so that the shortest note has a value of 1 in our notation. In most music an eighth note is the shortest note, so the time values translate like this:

The best way to check whether you have translated the music correctly is to add up all the time values in a DATA statement to see if they add up to eight for a 4/4 measure or six for a 3/4 measure.

Harmony

The harmony that we spoke of earlier is a scientific description of the agreement or "consonance" of two or more united sounds. The term also applies to a group of notes sounding together which are used as a background for the melody. Our two voice player has only one note for the harmony, but the four voice player has a group of notes called a chord that plays in the background. The harmony and chords generally are carried for several notes of the melody.

Notes that are played together are strung together on the same stem.

Harmony usually plays continuously in the songs; therefore the notes on the base clef are not shut off by the following sound statement in the player programs:

```
590 SOUND Ø,Ø,Ø,Ø,Ø.
```

The harmony notes continue until they are replaced by another note or a rest. Even if the harmony notes are held, the READ statement has to read all the notes for the melody, time and harmony each time it is executed. For example, the following READ statement has to match the number of elements of the DATA statement even though the harmony notes are the same for the melody notes 81,4 and 72,2. The harmony will continue to play while the melody notes change.

	First Pass	Second Pass
READ	NT, TM, A, B, C	NT, TM, A, B, C,
DATA	81, 4, 128, 162, 217,	72, 2, 128, 162, 217

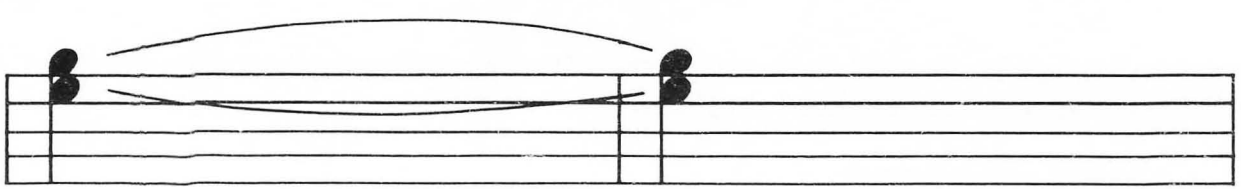
Writing the music takes a little longer when the harmony has to be repeated for each note, but the notation is very regular and easy to follow. When you play the music the PRINT statements will print out the note, time, and harmony for each note as it is played. If you hear a sour note or get an error message of some kind, the trouble is usually easy to spot.

The most common errors are:

- Leaving out a comma between numbers.
- Using a period instead of a comma.
- Putting in a comma after the word DATA or after the last DATA element.
- Leaving out or adding an extra number

If the music hangs on one note it indicates that the player has read a large note number for a time value, and is holding a note for a long time. If a section of the music mysteriously disappears or is out of place you may have repeated a line number. If you reuse a line number, the new DATA will replace the old DATA at that line number. This can easily happen with numbers like 1010 and 1110.

The following example of a slurred note indicates that the note continues:




NOW THINGS GET REALLY INTERESTING!

When music is rewritten to fit the capabilities of a certain instrument, the versions are called "arrangements." When you get vocal, guitar, and piano arrangements and translate them to Atari players there is no way to predict how good the music will sound. Try the melody by itself first and then add one voice of harmony. If that sounds good try four voices. Experiment until you get what you want.

7

THE ROUND PLAYER

Using Arrays

round is a melody that has voices entering at regular periods on the same note repeating the melody. "Row, Row, Row Your Boat" is the most popular example. The principle of the round is that the repeated note provides the harmony for the melody. The melody is, in effect, its own harmony. We have included rounds in this book because a special player was written just for rounds that will teach you a great deal about the principles of BASIC computing. The players we used for the music in the first part of this book read the music straight through from a data table and play the music from beginning to end. The RESTORE command is used to re-read the data. This system is not flexible enough for us to play rounds. In order to play the song over again in four voices we have to read the notes in four different places at once. The BASIC function that allows us to accomplish this magic is called an array. An array is like a wall of post office boxes. The array variable is the zip code of the post office, and the array element is the box number. The variable that we used to hold a single note in the player program can be assigned a series of numbers representing the array elements. Each element can hold a note value. You could number the variable NT from NT(0) to NT(1000). The Atari could have 1000 notes in its memory at one time and still not come close to using up the memory capacity. The number in parentheses can also be a variable NT(A), so we can "step" through the array as you will see.


ROW ROW ROW YOUR BOAT...
MERRILY MERRILY MERRILY MERRILY...
U.S. POST OFFICE

```
100 REM
110 REM  ****************************************
120 REM *                                        *
130 REM *          The Musical Atari             *
140 REM *                                        *
150 REM *          ROUND PLAYER                  *
160 REM *                                        *
170 REM *          by Hal Glicksman              *
180 REM *                                        *
190 REM *       © 1983 DATAMOST Inc.             *
200 REM *                                        *
210 REM  ****************************************
220 REM
230 DIM A(200)
240 REM *** DIMENSION AN ARRAY TO HOLD NOTES
250 PRINT "↰":REM CLEAR SCREEN COMMAND
260 POSITION 2,8
270 REM POKE 752,1:REM INVISIBLE CURSOR
280 PRINT "                    "
290 PRINT "                    "
300 PRINT "                    "
310 PRINT "                    "
320 PRINT "                    "
330 PRINT "                    "
340 NT=0:REM NOTE VARIABLE
350 B=1:REM ARRAY VARIABLE DURING PLAY
360 Z=1:REM ARRAY VARIABLE DURING READ
370 F=24:REM OFFSET OF ROUND REPEAT
380 REM *** SET VARIABLES TO ZERO
390 READ NT
400 REM *** READ NOTES FROM DATA STATEMENTS
410 A(Z)=NT
420 REM *** ASSIGN NOTE TO ARRAY ELEMENT Z
430 Z=Z+1
440 REM *** ADD ONE TO ARRAY ELEMENT
450 IF Z=196 THEN GOTO 490
```

```
┌──────────────┐
│              │
│ Row Row Row  │
│ Your Boat    │
│              │
└──────────────┘
```

```
460 REM *** GO TO PLAY SECTION AFTER NOTES HAVE FILLED ARRAY
470 GOTO 390
480 REM *** RETURN FOR ANOTHER NOTE
490 SOUND 0,A(B),10,10
500 REM *** PLAY FIRST VOICE
510 IF B>F THEN SOUND 1,A(B-F),10,8
520 REM *** SECOND VOICE
530 IF B>(2*F) THEN SOUND 2,A(B-(2*F)),10,6
540 REM *** THIRD VOICE
550 IF B>(3*F) THEN SOUND 3,A(B-(3*F)),10,4
560 REM *** FOURTH VOICE
570 FOR PLAY=1 TO 60:NEXT PLAY
580 REM *** HOLD NOTE
590 B=B+1
600 REM *** GET NEXT ARRAY ELEMENT
610 IF A(B)=1 THEN SOUND 0,0,0,0
620 REM *** END NOTE IF NUMBER ONE IS READ
630 IF B>F THEN IF (A(B-F))=1 THEN SOUND 1,0,0,0
640 REM *** CHECK VOICE TWO END
650 IF B>(2*F) THEN IF (A(B-(2*F)))=1 THEN SOUND 2,0,0,0
660 REM *** CHECK VOICE THREE END
670 IF B>(3*F) THEN IF (A(B-(3*F)))=1 THEN SOUND 3,0,0,0
680 REM *** CHECK VOICE FOUR END
690 B=B+1
700 REM *** GET NEXT ARRAY ELEMENT
710 IF B>165 THEN END
720 REM *** END WHEN DONE WITH NOTES
730 GOTO 490
740 REM *** RETURN FOR MORE NOTES
1000 REM
1010 REM ****************************************
1020 REM *                                      *
1030 REM *          TABLE FOR NOTES             *
1040 REM *            ROW ROW ROW               *
1050 REM *              YOUR BOAT               *
```

```
1060 REM *                                    *
1070 REM ************************************
1080 REM
1090 DATA 121,0,121,0,121,1,121,0,121,0,121,1
1100 DATA 121,0,121,1,108,1,96,0,96,0,96,1
1110 DATA 96,0,96,1,108,1,96,0,96,1,91,1
1120 DATA 81,0,81,0,81,1,0,81,0,81,0,1
1130 DATA 60,1,60,1,60,1,81,1,81,1,81,1
1140 DATA 96,1,96,1,96,1,121,1,121,1,121,1
1150 DATA 81,0,81,1,91,1,96,0,96,1,108,1
1160 DATA 121,0,121,0,121,0,121,0,121,0,121,1
1200 REM *** TABLE OF ZEROS ***
1210 DATA 0,0,0,0,0,0,0,0,0,0,0,0
1220 DATA 0,0,0,0,0,0,0,0,0,0,0,0
1230 DATA 0,0,0,0,0,0,0,0,0,0,0,0
1240 DATA 0,0,0,0,0,0,0,0,0,0,0,0
1250 DATA 0,0,0,0,0,0,0,0,0,0,0,0
1260 DATA 0,0,0,0,0,0,0,0,0,0,0,0
1270 DATA 0,0,0,0,0,0,0,0,0,0,0,0
1280 DATA 0,0,0,0,0,0,0,0,0,0,0,0
1290 DATA 0,0,0,0,0,0,0,0,0,0,0,0
```

How the Round Player Works

The notes for the round are read from DATA statements in line 390. The first note is placed in array A, element Z in line 410. Z started out with the value of 1, so A (1) is the first note. Z has 1 added to it in line 430 and the program is sent around for another note. Z is now 2, so the second note will be stored in A (2). If variables make your head spin type in the first part of the program up to line 470 and type in a few lines of the DATA statements starting at line 800. This will be enough the run the array part of the program.

After you RUN the program you will be able to print out the notes from the array. Even if you have typed in the entire program and played the music, the array will still be filled with notes. Type:

```
PRINT A(1)
121
```

You can even put a FOR/NEXT loop on a single line in immediate mode to print out the entire array:

```
FOR I=1 TO 196:PRINT A(I),:NEXTI
```

The time value of the round cannot be written the same way as the previous players, because a note in one voice might be held while another voice changes notes. The notation is written with a note value for each beat. The value following the note is either 0 if the note is held for another beat or 1 if the note ends on that beat. The following DATA statement represents the data for two C notes held for three beats each:

```
DATA 121,Ø,121,Ø,121,1,121,Ø,121,Ø,121,1
```

After all the notes for the round are entered, a large number of 0's are required to fill up the notes of the voices that are finished playing as the round trails off. It doesn't matter if there are more than enough 0's, so write out line 1210 and re-number this line 1220, 1230, etc. until you have more lines of 0's than lines of notes.

The first section of the program will continue to read notes until Z = 196, then the program will branch to the second section where the notes are played in 490 to 730. The second voice starts playing 24 notes after the first voice as follows:

```
51Ø IF B>F THEN SOUND 1,A(B-F),1Ø,8
```

F is set to the number of beats that the second voice trails behind the first voice (24). If you subtract 24 from B when the music has just started, you will get a negative result, because the music has not played past the first 24 notes. In order to avoid negative values for B we have a test in the beginning of the statement: IF B>F. This test keeps the second voice from playing until B is past the first 24 notes. The third and fourth voice work the same way as the second voice with the music coming in at 2 times F and 3 times F.

Line 590 adds 1 to B, and that will always be the time value of 0 or 1.

0=hold the note
1=end the note

Lines 600 to 680 check to see if the time value is 1 in order to turn off the appropriate voice and end the note.

```
63Ø IF B>F THEN IF (A(B-F))=1 THEN SOUND 1,Ø,Ø,Ø
```

This line has two IF THEN tests in a row. The first tests whether the song has played past the first 24 notes. The second tests whether B equals 1 in order to turn off the note.

In line 690 a 1 is added to B again to move to the next array element that holds a note value. Line 730 sends the program around to play the next note. This is what the array look slike after it has read the data from the data statements:

A(1)	A(2)	A(3)	A(4)	A(5)	A(6)	A(7)	A(8)	A(9)	A(10)..... etc.
121,	0,	121,	0,	121,	1,	121,	0,	121,	0 ..... etc.

As the program steps through the array the notes are played in the following way:

- B counts the array elements; it is set to 1 in line 350. A(B) is equal to A(1).
- A(B) in line 490 plays the note that is stored as the first array element.
- In line 590, 1 is added to variable B to read the next array element, the time value.
- Because B is now equal to 2, A(B) is equal to A(2). The array element A(2) is a time value that is used in the program in line 610.
- This process is continued as the program loops around:

```
B=B+1
B=3
A(B)=A(3)   [a note value]

B=B+1
B=4
A(B)=A(4)   [a time value]

etc.
```

Since the array keeps track of all the notes and times, the round player can be playing four different parts of the music at once. Each round melody requires a matching array size, and the repeat may start in a different place. These values are noted along with the notes for each round in Chapter 13. We admit that rounds are very complicated, but the program demonstrates important capabilities of your Atari home computer.

8

THE ATARI KEYBOARD ORGAN

Playing Live Music on the Atari

ou can use your Atari to play music just like one of the small electronic keyboards with one and a half octaves. Our keyboard organ program has six chords, but there is plenty of room on the keyboard for more chords or special effects if you want them. The purpose of the keyboard organ in this book is to allow you to play tunes "by ear" and get a display of the Atari SOUND values on the screen. The values for the notes and the chords can then be entered into the data table of the four voice player. The keyboard organ actually makes the Atari keyboard into an electronic organ of surprising quality and playability.

```
100 REM
110 REM  ****************************************
120 REM *                                        *
130 REM *     THE ATARI KEYBOARD ORGAN           *
140 REM *                                        *
150 REM *          BY HAL GLICKSMAN              *
160 REM *                                        *
170 REM *        © 1983 DATAMOST INC             *
180 REM *                                        *
190 REM  ****************************************
200 REM
210 HM=3
220 REM *** SET LOUDNESS OF CHORDS
230 PRINT "↶":REM CLEAR SCREEN
```

```
240 DIM A(100)
250 REM SET UP ARRAY FOR NOTE VALUES
260 REM **********
270 FOR I=0 TO 99
280 READ NT
290 A(I)=NT
300 NEXT I
310 REM *********
320 PRINT "                    PLAY!!"
330 OPEN #1,4,0,"K:"
340 REM *** OPEN KEYBOARD FOR READING
350 KEY=PEEK(764)
360 REM *** GET NOTE BY PEEKING KEYBD
370 IF KEY=HAL THEN SOUND 0,0,0,0
380 REM *** IF NOTE IS THE SAME THEN TURN OFF SOUND
390 IF KEY=255 THEN GOTO 350
400 REM *** GO BACK TO READ KEY AGAIN
410 REM ********************
420 REM TEST FOR KEYS 1 TO 7
430 IF KEY=31 THEN GOSUB 670
440 IF KEY=30 THEN GOSUB 720
450 IF KEY=26 THEN GOSUB 770
460 IF KEY=24 THEN GOSUB 820
470 IF KEY=29 THEN GOSUB 870
480 IF KEY=27 THEN GOSUB 920
490 IF KEY=51 THEN GOSUB 970
500 REM ********************
510 REM *** GO TO CHORDS
520 IF KEY>99 THEN KEY=99
530 SOUND 0,A(KEY),10,10
540 REM *** PLAY MELODY NOTES
550 PRINT "ATARI SOUND=";A(KEY)
560 REM *** PRINT OUT NOTE VALUES
570 HAL=KEY
580 REM *** STORE LAST KEY PRESSED
590 POKE 764,255
```

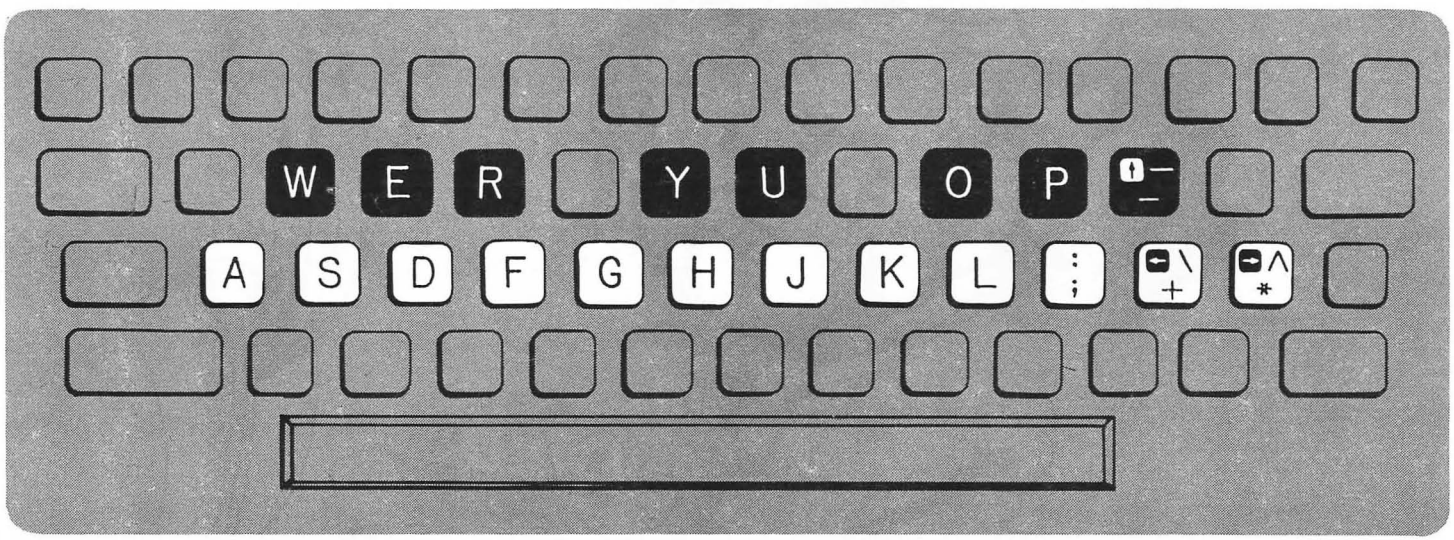
```
600 REM *** EMPTY KEY READER
610 GOTO 350
620 REM GET ANOTHER NOTE
630 REM ******
640 REM CHORDS
650 REM ******
660 REM *** CHORD I
670 SOUND 1,121,10,HM
680 SOUND 2,96,10,HM
690 SOUND 3,81,10,HM
700 RETURN
710 REM *** CHORD IV
720 SOUND 1,182,10,HM
730 SOUND 2,144,10,HM
740 SOUND 3,121,10,HM
750 RETURN
760 REM *** CHORD V
770 SOUND 1,128,10,HM
780 SOUND 2,91,10,HM
790 SOUND 3,81,10,HM
800 RETURN
810 REM *** CHORD II
820 SOUND 1,108,10,HM
830 SOUND 2,91,10,HM
840 SOUND 3,72,10,HM
850 RETURN
860 REM *** CHORD III
870 SOUND 1,96,10,4
880 SOUND 2,81,10,4
890 SOUND 3,64,10,4
900 RETURN
910 REM *** CHORD VI
920 SOUND 1,144,10,4
930 SOUND 2,121,10,4
940 SOUND 3,96,10,HM
```

```
950 RETURN
960 REM *** TURN OFF HARMONY
970 SOUND 1,0,0,0
980 SOUND 2,0,0,0
990 SOUND 3,0,0,0
1000 RETURN
1010 REM
1020 REM ****************************************
1030 REM *                                      *
1040 REM *        TABLE OF NOTE VALUES          *
1050 REM *                                      *
1060 REM *     AND NUMERIC EQUIVALENTS          *
1070 REM *                                      *
1080 REM ****************************************
1090 REM
1100 REM 0-9
1110 DATA 081,096,072,000,000,091,064,060,085,000
1120 REM 10-19
1130 DATA 076,102,000,000,068,000,000,000,000,000
1140 REM 20-29
1150 DATA 000,000,000,000,000,000,000,000,000,000
1160 REM 30-39
1170 DATA 000,000,000,000,000,000,000,000,000,000
1180 REM 40-49
1190 DATA 136,000,153,114,000,000,173,000,000,000
1200 REM 50-59
1210 DATA 000,000,000,000,000,000,128,108,144,000
1220 REM 60-69
1230 DATA 000,121,162,182,000,000,000,000,000,000
1240 REM 70-79
1250 DATA 000,000,000,000,000,000,000,000,000,000
1260 REM 80-89
1270 DATA 000,000,000,000,000,000,000,000,000,000
1280 REM 90-99
```

```
1290 DATA 000,000,000,000,000,000,000,000,000,000
1300 DATA 000,000,000,000,000,000,000,000,000,000
```

PEEKing at the Keyboard

It is often a revelation to new computerists to discover that the computer keyboard is not "locked-in" to the letters that are printed on top of the keys. In one program a certain key can draw a line and in another program the same key can print a Japanese character or a math symbol. The keyboard organ program "maps" a piano keyboard onto the keyboard in the pattern shown here:



You can easily arrange the keyboard in any pattern you desire. The note values are assigned to the keys by an array table created in lines 260 to 300. After the array has read the data from the data table, the word "PLAY!!" comes on the screen and the keyboard plays. The keyboard is "read" by PEEKing at the memory location that stores value of the last key pressed.

```
350 KEY = PEEK (764)
```

This command is very useful. Location 764 stores the value of the last key pressed until a new key is pressed. The location can also be cleared by POKEing 255 into that location. You do not need to know what PEEKs and POKEs actually do to make use of them in programs. The number that KEY retrieves from the

keyboard is not the ATASCII code that is charted in the Atari BASIC Programming Manual, but is a hardware code deep inside the machine. You don't need to consult a chart, just ask the computer what the code is with this little program:

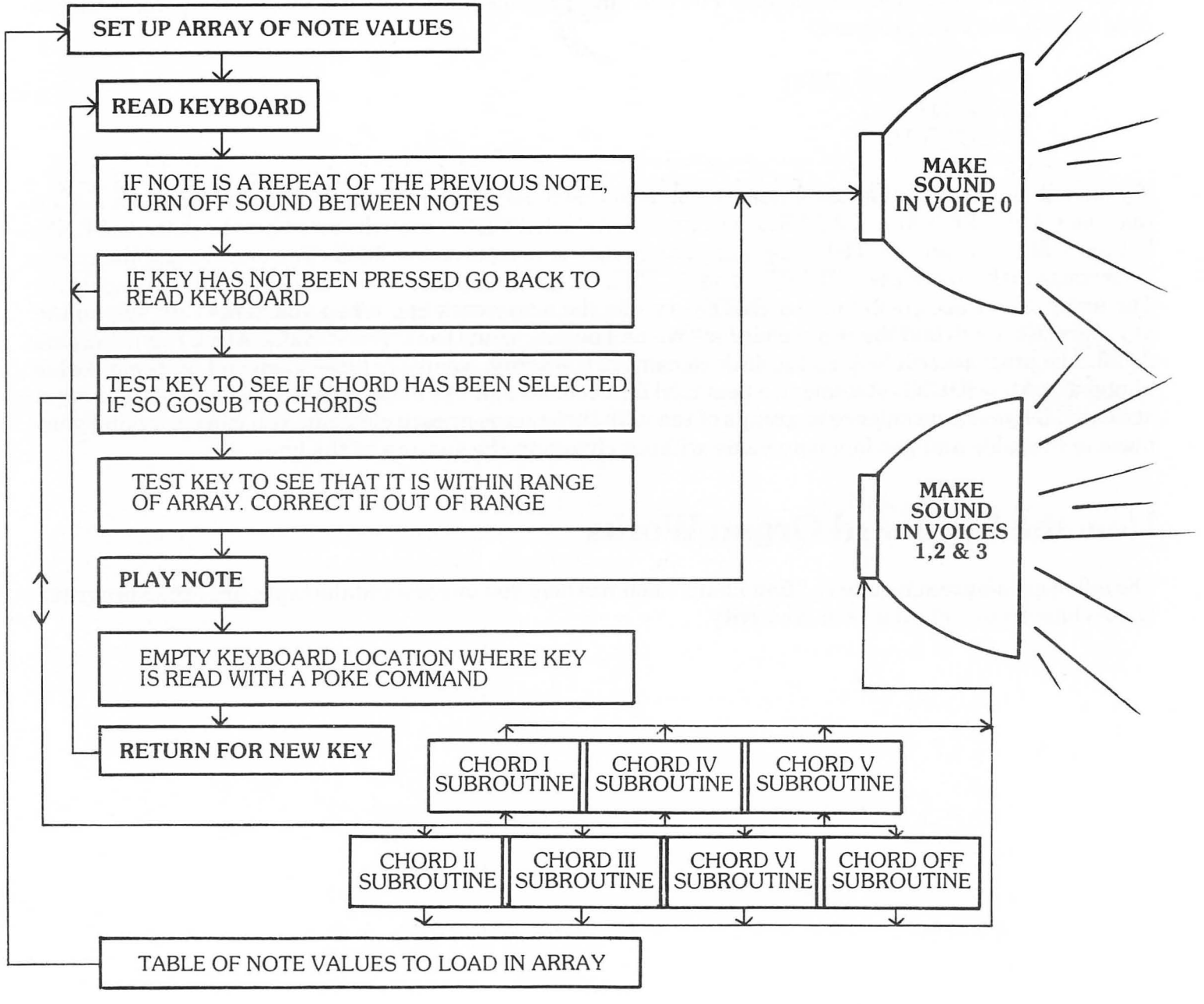
```
10 KEY=PEEK(764)
20 PRINT KEY
30 GOTO 10
```

If you want to change the key assignments just write down the numbers for the keys you want to use. Notice that the <ESCAPE> key, <ATARI> key and <CAPS/LOWER> keys do not effect the output of KEY, but the <SHIFT> and <CTRL> keys do change the value returned by KEY. You can expand the organ program to make use of the <SHIFT> and <CTRL> keys to play another octave or another musical key. The array "A" is set up so that "A (KEY)" will be the array variable. When you press the "W" on the keyboard, KEY will find the secret code for "W" and become equal to 46. The variable A(KEY) will become A(46). The program will look up the 46th element in the array, where we have stored a 173. When 173 is plugged into the SOUND statement, a beautiful B flat comes out. The data table that makes up the array is arranged for your convenience in groups of ten with three zeros in each element. You can easily find your place in the table and put in a note value without changing the spacing of the lines.

How the Keyboard Organ Works

The following diagram is called a "flow chart," and will help you understand the keyboard organ program. Make changes to suit your own creativity.

Flowchart for Keyboard Organ



9

SPECIAL EFFECTS WITH SOUND

tari sound effects can be used very effectively in games and other programs to suggest real life sounds. The human ear is so sensitive, however, that even the best synthesizers sound unreal. A good part of any illusion is suggestion. If you see a car on the screen and hear a roaring sound, you will be more likely to accept the sound as a car engine than you would if you heard the same sound with nothing on the screen to look at. The short sound routines in this section will be most effective if they are worked into a game or used as sound effects for a text adventure.

If you multiply all the sound values times the possible distortions, loudness values and voices, the resulting number of possible sounds is huge. In addition to all these sounds, programs which make the sounds flutter or rise and fall can add to the number of possible sound effects. The program below gives you a quick way of exploring the range of sounds possible with different sound values and distortions. The joystick controls pitch, distortion and volume. The trigger button switches the joystick control from one voice to another. A screen display in large letters tells you what values you are listening to and what voice is being changed.

```
100 REM ****************************************
110 REM *                                      *
120 REM *          ATARI SOUND MIXER           *
130 REM *                                      *
140 REM ****************************************
150 REM
160 REM ****************************************
170 REM *     SET UP ARRAYS WITH ZEROS         *
180 REM ****************************************
```

```
190 DIM P(4),D(4),V(4)
200 FOR A=1 TO 4:P(A)=0:D(A)=10:V(A)=8:NEXT A
210 REM *** SET UP LARGE PRINT
220 GRAPHICS 18:PRINT #6;"  SOUND MIXER"
230 REM
240 GOSUB 330
250 X=1:GOTO 400
260 REM
270 REM  ****************************************
280 REM *         POSITION PRINT OUT FOR         *
290 REM *            SOUND VALUES AND            *
300 REM *                PRINT ZEROS             *
310 REM  ****************************************
320 REM
330 N=0:FOR S=3 TO 9 STEP 2:POSITION 1,S:PRINT #6;"SOUND ";
    N:N=N+1:NEXT S:RETURN
340 REM
350 REM  ****************************************
360 REM *          READ PADDLE BUTTON            *
370 REM *            POSITION PRINTING           *
380 REM *                                        *
390 REM  ****************************************
400 IF STRIG(0)=0 THEN GOSUB 330:X=X+1:GOTO 540
410 REM
420 REM  ****************************************
430 REM *        READ PADDLE AND CHECK           *
440 REM *            TO SEE THAT VALUES          *
450 REM *          ARE NOT OUT OF RANGE          *
460 REM  ****************************************
470 REM
480 IF STICK(0)=13 THEN P(X)=P(X)+1:IF P(X)>255 THEN P(X)=0
490 IF STICK(0)=14 THEN P(X)=P(X)-1:IF P(X)<0 THEN P(X)=255
500 IF STICK(0)=11 THEN D(X)=D(X)+2:FOR I=1 TO 30:NEXT I:IF D(X)>14
    THEN D(X)=0
```

```
510 IF STICK(0)=7 THEN V(X)=V(X)+1:FOR I=1 TO 30:NEXT I:IF V(X)>15
    THEN V(X)=0
520 REM
530 REM *** RESET X TO TOP OF SCREEN
540 IF X>4 THEN X=1
550 REM
560 REM *** PRINTS 'SOUND' IN BLUE
570 POSITION 1,X+X+1:PRINT #6;"sound"
580 REM
590 REM *** PRINTS VALUES FOR SOUND
600 POSITION 8,X+X+1:PRINT #6;",";P(X);",";D(X);",";V(X);"  "
610 REM
620 REM *** PLAY SOUND FOR 4 X VALUES
630 SOUND X-1,P(X),D(X),V(X):GOTO 400
```

The GRAPHICS 18 command sets the screen for large text printing. In this mode the capital letters in the program are printed in yellow, and the lower case letters in the program appear on the screen as blue upper case letters. This effect is used in line 560 to make the active SOUND command light up in blue on the screen. The only other line that is a little bit complex is line 330. This is a FOR/NEXT loop that clears out the SOUND value printouts and prints a yellow "SOUND" in the line to erase the blue printing. The sound mixer program is good for testing the SOUND statement possibilities when you are working out your own sound effects.

Joystick Space Zap

If you want to create eerie outer space effects with the joystick the program below is just what you need. It was inspired by one of the first electronic musical instruments called the Theramin, which dates back to the early days of radio. The Theramin provided spooky background music for countless horror movies and radio mysteries. The program is called Joystick Space Zap, and it creates just the kind of spooky sounds you would expect from a computer. The program is explained by the REM statements.

```
100 REM  ****************************************
110 REM *                                       *
120 REM *          JOYSTICK SPACE ZAP           *
```

```
130 REM *                                     *
140 REM *          BY HAL GLICKSMAN           *
150 REM *                                     *
160 REM *        © 1983 DATAMOST INC          *
170 REM *                                     *
180 REM  *************************************
190 REM
200 REM *** SET BEGINNING VALUES
210 REM
220 A=121
230 B=121
240 C=2
250 REM *** CLEAR SCREEN
260 PRINT CHR$(125)
270 REM *** SET LARGE PRINT GRAPHICS
280 GRAPHICS 18
290 PRINT #6;"    SPACE ZAPPER"
300 REM *** READ JOYSTICK VALUE
310 X=STICK(0)
320 REM
330 REM  *************************************
340 REM *       ADD OR SUBTRACT FROM          *
350 REM *            SOUND VALUES             *
360 REM  *************************************
370 REM
380 IF X=14 THEN A=A-1
390 IF X=14 THEN B=B-1
400 IF X=10 THEN A=A-1
410 IF X=11 THEN C=C-1
420 IF X=9 THEN A=A-1
430 IF X=13 THEN A=A+1
440 IF X=13 THEN B=B+1
450 IF X=5 THEN B=B+1
460 IF X=7 THEN C=C+1
470 IF X=6 THEN B=B-1
```

```
480 IF A<1 THEN A=1
490 REM
500 REM  ****************************************
510 REM *         CHECK THAT VALUES ARE          *
520 REM *       IN THE ALLOWABLE RANGE           *
530 REM  ****************************************
540 REM
550 IF A>255 THEN A=255
560 IF B<1 THEN B=1
570 IF B>255 THEN B=255
580 IF C<1 THEN C=1
590 IF C>15 THEN C=15
600 REM
610 REM *** PRINT VALUES
620 REM
630 POSITION 1,3
640 PRINT #6;"VOICE ONE NOTE=";A;" "
650 POSITION 1,6
660 PRINT #6;"VOICE TWO NOTE=";B;" "
670 POSITION 1,9
680 PRINT #6;"JOYSTICK VALUE=";X;" "
690 REM
700 REM *** PLAY SOUNDS
710 REM
720 SOUND 0,A,10,C
730 SOUND 1,B,10,C
740 REM
750 REM *** RETURN FOR NEW NOTE
760 REM
770 GOTO 310
```

Fade Routines

A fade routine is a couple of lines added to a program to make the note trail off instead of ending abruptly. The first two special effects are simply demonstrations of how "fast fade" and "slow fade" can enrich Atari

sound. The Zip Gun Shot and Fill the Bottle sound effects both use fade techniques.

```
100 REM  ****************************************
110 REM *                FAST FADE                *
120 REM  ****************************************
130 REM
140 FOR REPEAT=1 TO 10
150 FOR I=24 TO 1 STEP -1
160 SOUND 0,121,10,I/2
170 NEXT I
180 FOR PAUSE=1 TO 20
190 NEXT PAUSE
200 NEXT REPEAT
```

```
100 REM  ****************************************
110 REM *                SLOW FADE                *
120 REM  ****************************************
130 REM
140 FOR I=100 TO 1 STEP -1
150 SOUND 0,121,10,I/10
160 NEXT I
```

Fade Routine for the Player Programs

Fade techniques may be entered into the player programs by changing the "hold the note" section in line 530 of the player programs. You can add this effect to any song that already has a player; it works with both the two and four voice versions. Be sure a correct version of your song and player is saved on disk or cassette. Then you can tinker safely with the version that is currently loaded into your Atari.

Delete line 480 by typing the following:

```
480<RETURN>
```

Enter these lines:

```
530 FOR I=15 TO 4 STEP -1
532 FOR PLAY = 1 TO TM*3:NEXT PLAY
534 SOUND 0,TM,10,I
536 NEXT I
540 REM FADE ROUTINE
```

Try the player this way. Also try the harmony sound volumes reduced from 4 to 2. This version of the player sounds very artificial and may not be to your liking, but it does demonstrate that the player can be tinkered with to change the sound, and will in turn change the sound of every song it plays.

Sound Effects Programs

Sound effects are simple programs that use FOR/NEXT loops to run through different note, distortion or volume values. The only command that is unusual is the ABS command in the Volcano and Siren sounds. ABSolute means ignore the – sign in front of any number and treat the number as a positive value. If the program counts from –10 to 10 and then takes the ABS of the result, the output will be:

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

When you use this routine with the SOUND statement, the computer turns the volume down and back up in one smooth operation.

```
100 REM  ****************************************
110 REM *                VOLCANO                  *
120 REM  ****************************************
130 REM
140 PRINT "Duration:";:INPUT X
150 FOR J=-10 TO 10
160 SOUND 0,200,4,12-ABS(J)
170 SOUND 1,255,4,12-ABS(J)
180 SOUND 2,225,4,12-ABS(J)
190 SOUND 3,150,4,12-ABS(J)
```


RUMMBBLLLEEE...
SOUNDS A BIT OUT OF TUNE TODAY...
--THINK WE SHOULD DO SOMETHING?
MAYBE. YOUR DAUGHTER GET MARRIED YET?
NAAHHH---

```
200 FOR K=1 TO X:NEXT K
210 NEXT J:END

100 REM  ****************************************
110 REM *                JET TAKE OFF              *
120 REM  ****************************************
130 REM
140 FOR L=1 TO D
150 FOR J=0 TO 45
160 SOUND 0,J,8,J/3
170 NEXT J
180 FOR J=45 TO 0 STEP -1
190 SOUND 0,J,8,J/6
200 FOR K=1 TO 70+J*3:NEXT K
210 NEXT J:NEXT L:END

100 REM  ****************************************
110 REM *                ZIP GUN SHOT              *
120 REM  ****************************************
130 REM
140 PRINT "How many shots?";:INPUT D
150 FOR J=1 TO D
160 SOUND 0,5,0,15
170 SOUND 1,6,0,15
180 FOR K=1 TO 18:NEXT K
190 SOUND 0,0,0,0
200 FOR T=15 TO 0 STEP -1
210 SOUND 1,0,0,T
220 NEXT T
230 FOR K=1 TO 100:NEXT K
240 NEXT J:END

100 REM  ****************************************
110 REM *              TELEPHONE RING              *
120 REM  ****************************************
```

```
130 REM
140 PRINT CHR$(125):REM CLEAR SCREEN
150 PRINT "How many rings?:";:INPUT D
160 FOR J=1 TO D
170 SOUND 0,66,10,5
180 SOUND 1,68,10,5
190 SOUND 2,120,2,4
200 FOR K=1 TO 500:NEXT K
210 FOR K=0 TO 3:SOUND K,0,0,0:NEXT K
220 IF J=D THEN GOTO 240
230 FOR K=1 TO 750:NEXT K
240 NEXT J
250 GOTO 140
```

```
100 REM  ***************************************
110 REM *          THE TWITTER MACHINE          *
120 REM  ***************************************
130 REM
140 FOR HAL=1 TO RND(1)*20
150 FOR BIRD=4 TO 12
160 SOUND 0,BIRD,10,10
170 NEXT BIRD
180 NEXT HAL
190 SOUND 0,0,0,0
200 FOR PAUSE=1 TO RND(1)*200
210 NEXT PAUSE
220 FOR HAL=1 TO RND(1)*15
230 FOR BIRD=4 TO 12
240 SOUND 0,BIRD,10,12-BIRD
250 NEXT BIRD
260 SOUND 0,0,0,0
270 NEXT HAL
280 FOR PAUSE=1 TO RND(1)*20
290 NEXT PAUSE
300 GOTO 140
```

```
100 REM  ***************************************
110 REM *        GALACTIC ZAPPER GIZMO          *
120 REM  ***************************************
125 REM
130 FOR HAL=1 TO RND(1)*15
140 FOR ZAP=4 TO 12
150 SOUND 0,ZAP,10,12
160 SOUND 0,ZAP+ZAP,10,10
170 NEXT ZAP
180 SOUND 0,0,0,0
190 NEXT HAL
200 FOR PAUSE=1 TO RND(1)*20
210 NEXT PAUSE
220 GOTO 130
```

```
100 REM  ***************************************
110 REM *                JACK HAMMER             *
120 REM  ***************************************
130 REM
140 FOR BANG=1 TO 45
150 SOUND 0,84+RND(0)*3,2,10
160 FOR PAUSE=1 TO 40+RND(0)*50:NEXT PAUSE
170 NEXT BANG
180 END
```

```
100 REM  ***************************************
110 REM *                   SIREN                *
120 REM  ***************************************
130 REM
140 FOR NUM=1 TO 12
150 FOR TONE=-60 TO 60 STEP 2
160 SOUND 0,ABS(TONE)+80,10,10
170 FOR WAIT=1 TO 8
180 NEXT WAIT
190 NEXT TONE
```

```
200 NEXT NUM
210 END

100 REM  ****************************************
110 REM *              ENGLISH SIREN              *
120 REM  ****************************************
130 REM
140 FOR NUM=1 TO 10
150 SOUND 0,50,14,10
160 SOUND 1,54,10,10
170 FOR WAIT=1 TO 100
180 NEXT WAIT
190 SOUND 0,66,10,8
200 SOUND 1,68,10,10
210 FOR WAIT=1 TO 100
220 NEXT WAIT
230 NEXT NUM
240 END

100 REM  ****************************************
110 REM *              ZOOPING SOUND              *
120 REM  ****************************************
130 REM
140 FOR NUM=1 TO 4
150 FOR I=255 TO 10 STEP -2
160 SOUND 0,I,10,10
170 SOUND 1,I-6,10,10
180 NEXT I
190 FOR PAUSE=1 TO 100
200 NEXT PAUSE
210 FOR I=10 TO 255 STEP 1
220 SOUND 0,I,10,10
230 SOUND 1,I-6,10,10
240 NEXT I
250 FOR PAUSE=1 TO 100
```


USE SOUND EFFECTS WISELY & WELL!
SPROING!
MEOW!
THOOOMMM---
BLAM!
BLAM!
BLAM!
BLAM!
BLAM!
BLAM!

```
260 NEXT PAUSE
270 NEXT NUM
280 END

100 REM ****************************************
110 REM *              FILL THE BOTTLE           *
120 REM ****************************************
130 REM
140 FOR DROP=150 TO 60 STEP -2
150 FOR FADE=30 TO 8 STEP -4
160 SOUND 0,DROP,10,FADE/2
170 NEXT FADE
180 FOR WAIT=1 TO 10:NEXT WAIT
190 SOUND 0,0,0,0
200 NEXT DROP
210 END
```

We urge you to tinker with these special effects programs. Your idea of the sound of a volcano may be very different from ours. The sound effects are short enough to put into other programs for dramatic effect. Since most BASIC games and programs have space between line numbers, you can insert these routines in many places, just as we put the fade routine in the player between existing line numbers. If you're not sure how the program that you are tinkering with works, put the sound routine at the end of the program. If you add RETURN as the last line of the sound routine it becomes a subroutine. Use trial and error to find the correct location in the program to add a GOSUB to your subroutine. If the GOSUB is in the wrong place it can be deleted and re-tried someplace else. Admittedly, this technique is very crude, but tinkering is a good way to learn about programming.

10

THE SONG MENU

f you have come this far in learning music and programming you might wish you could just look at a list of songs and press a key to play one. Well this is the computer age, and modern computer programs are "menu driven." This means that the choices available to you are presented on the screen in a list or menu. The program in this section allows you to put five songs in the player at once and create a little menu for them from which you can select one to play by pressing a single key. You could write a version that holds even more songs if you wish. What makes the menu possible is a special feature of the Atari RESTORE command that lets you point to a section of data to read. The menu, player, and the songs can all be assembled into a single program using the disk drive or program recorder.

```
5 REM *** MENU PROGRAM ***
10 OPEN #1,4,0,"K:"
15 DIM A$(6),B$(40),C$(40)
20 FOR I=0 TO 3:SOUND I,0,0,0:NEXT I
25 PRINT "↶":PRINT
30 PRINT "              THE MUSICAL ATARI"
35 PRINT :PRINT "                 SONG LIST"
40 POSITION 2,6
45 PRINT "1) On Top of Old Smokey"
50 PRINT :PRINT "2) Kumbaya"
55 PRINT :PRINT "3) I Know Where I'm Going"
60 PRINT :PRINT "4) For He's a Jolly Good Fellow"
65 PRINT :PRINT "5) My Bonnie Lies Over the Ocean"
```

```
70 PRINT :PRINT "WHAT NUMBER SONG WOULD YOU LIKE";
75 GET #1,PICK
80 PICK=PICK-48
85 ON PICK GOSUB 95,120,145,170,195
90 GOTO 220
95 RESTORE 2000
100 B$="                On Top of                "
105 C$="               Old Smokie                "
110 L=40
115 RETURN
120 RESTORE 3000
125 B$="                                         "
130 C$="                 Kumbaya                 "
135 L=40
140 RETURN
145 RESTORE 4000
150 B$="              I Know Where               "
155 C$="              I'm Going                  "
160 L=40
165 RETURN
170 RESTORE 5000
175 B$="            For he's a Jolly             "
180 C$="            Good Fellow                  "
185 L=40
190 RETURN
195 RESTORE 6000
200 B$="            My Bonnie Lies               "
205 C$="            Over the Ocean               "
210 L=40
215 RETURN
220 REM *** ADD FOUR VOICE PLAYER HERE ***
```

The first line of the menu program "opens" the keyboard for reading the song number you select. The OPEN command has to be used before the GET command will work. Line 20 shuts off the sound when the

program returns after playing a song. The menu program has a selection screen with the song titles in lines 25 to 70. Line 75 GETs the value of the number keys in ATASCII values; the codes that Atari uses to assign letters, numbers, and graphic characters. The numbers run from 48 for 0 to 57 for 9. When line 80 subtracts 48 from the value of PICK then PICK equals the number on the key pressed. The ON GOSUB command in line 85 goes to a subroutine that matches the song title selected. The subroutine selects the proper section of data by using the RESTORE command. B$ and C$ are assigned to the song titles and even have the edges of the title box included. Finally a value for L (tempo) is assigned for each song. After the program RETURNs from the subroutine, line 90 sends control to the player, and the song plays!

Combining Program Segments

It is not necessary to type the player or song notes again. The Atari allows you to put parts of programs together easily. Programs can be put together after they have been saved to disk or tape using the LIST command:

```
LIST"D:PROGRAM.LST",2000,2900 <RETURN>
```

This will save lines 2000 to 2900 of the program in computer memory under the name PROGRAM.LST. The LST part of the name is there to remind you that the file on the disk is a list, and not a BASIC program. When you want to put a listing back in computer memory type:

```
ENTER"D:PROGRAM.LST" <RETURN>
```

ENTER and LIST will allow you to save the song notes by themselves and put them together in another program. The menu and player programs can be assembled with the songs into one long program with a menu, a player, and five song tables.

When the menu program is typed into the computer, save it with the following command:

```
LIST"D:MENU.LST" <RETURN>
```

If you don't put line numbers after the file name, the computer will save the entire program that is currently in memory.

Now load in the FOURVOIC program or any song that uses the fourvoice player. A few changes are necessary before the player can be combined with the menu. The repeat feature is changed to a command to return to the menu, and the title screen is read from the menu. LIST the player program, and type these lines at the bottom of the screen. You do not have to type them where they belong in the program, the computer will automatically replace the old program lines with these new lines when you list the program.

```
290 REM
```

This eliminates the line where T (repeat) was set.

```
160 PRINT B$
370 PRINT C$
```

The song title from menu is printed in the title box.

```
400 REM
```

This eliminates the line where L (tempo) was set.

```
440 IF NT=1 THEN GOTO 20
```

This returns you to the menu. When the changes have been made, store the modified player using the LIST command.

```
LIST"FOURVOIC.LST",230,670 <RETURN>
```

This command will save the player program without the title panel and notes. Notice that the numbering scheme leaves room for the menu program to fit between 10 and 220.

Load one of the songs you want to have in the menu and renumber the data statements from 2000 up. Renumbering is very easy because the cursor jumps to the beginning of the next line every time you press <RETURN>. You don't have to delete the original data lines, because the LIST command will only save the lines you have numbered in the 2000 series.

```
LIST"D:SMOKEY.N2",2000,2999 <RETURN>
```


HEY-- I'LL SLIP YOU A TEN-SPOT TO PLAY THAT LAST NUMBER AGAIN--!
SURE THING, MAN--

This will save any lines in the 2000's. The ".N2" after the song name will remind you that the data is numbered in the 2000's.

Load another song and renumber the notes starting at 3000.

The menu player can now be assembled from the LISTed parts:

```
ENTER "D:MENU.LST" <RETURN>
ENTER "D:PLAYER.LST" <RETURN>
ENTER "D:SMOKEY.N2" <RETURN>
ENTER "D:[song name .N3" <RETURN>
ENTER "D:[song name .N4" <RETURN>
ENTER "D:[song name .N5" <RETURN>
ENTER "D:[song name .N6" <RETURN>
```

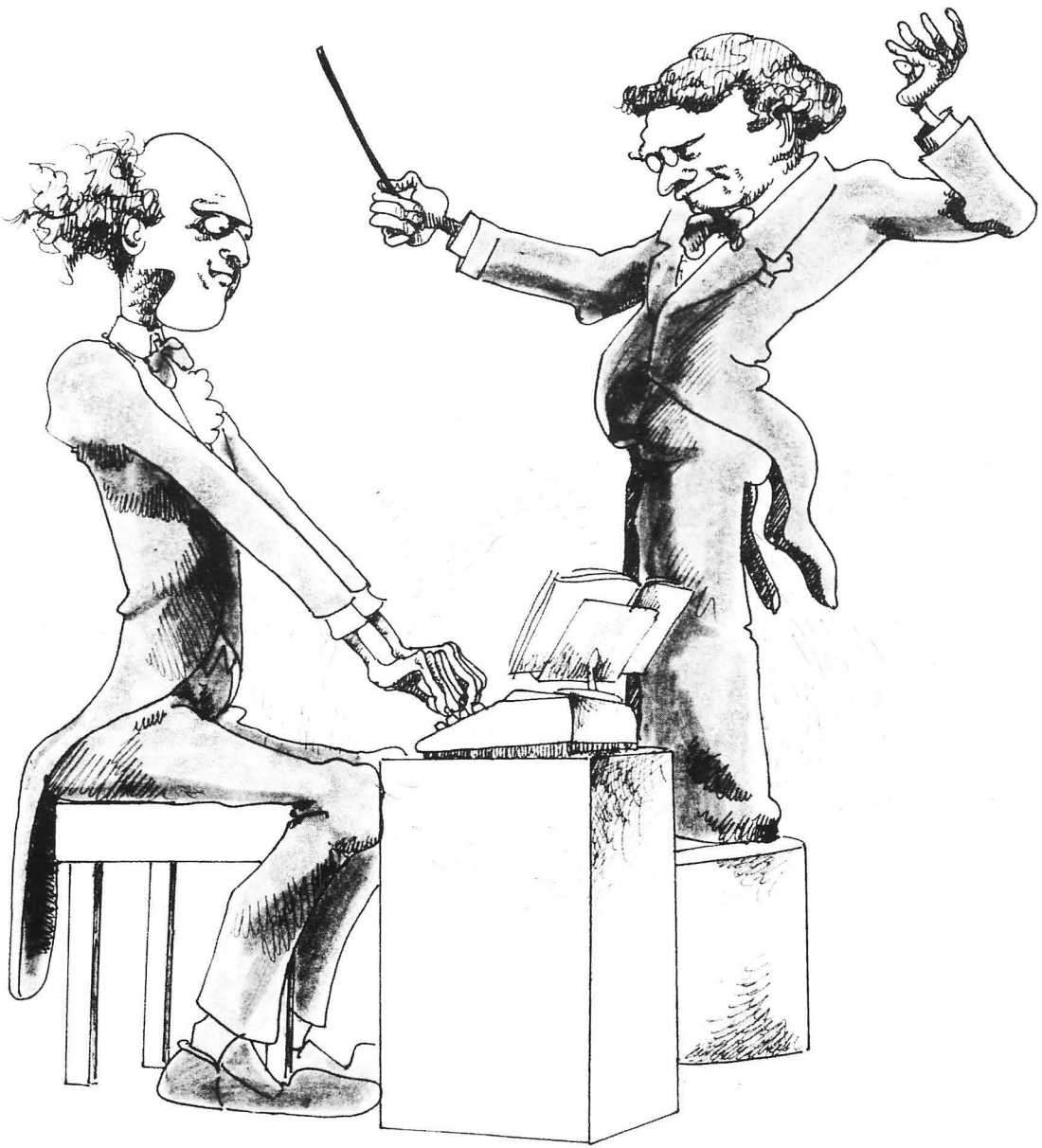
At Your Fingertips

When everything is in the Atari's memory the combined program should RUN with no problems. If there are errors in your program you should look carefully to see if some line in a song got renumbered with three digits instead of four digits, (such as 312 instead of 3120). A mistake like that could cause no end of mischief. Otherwise, if the songs and player worked by themselves, they should work in the menu.

Save the complete menu, player, and songs with the SAVE command:

```
SAVE"D:SONGMENU" <RETURN>
```

The assembled menu/player combination will be on disk as a BASIC program ready to play any time.



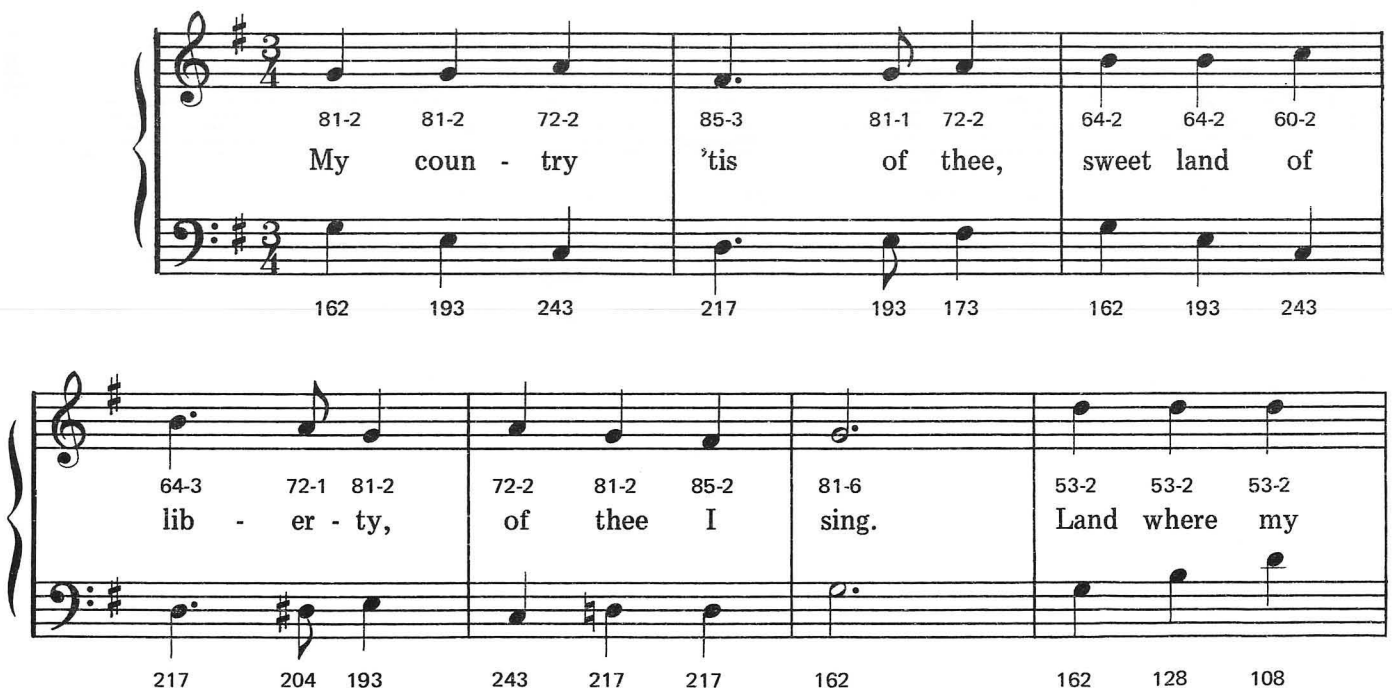
11

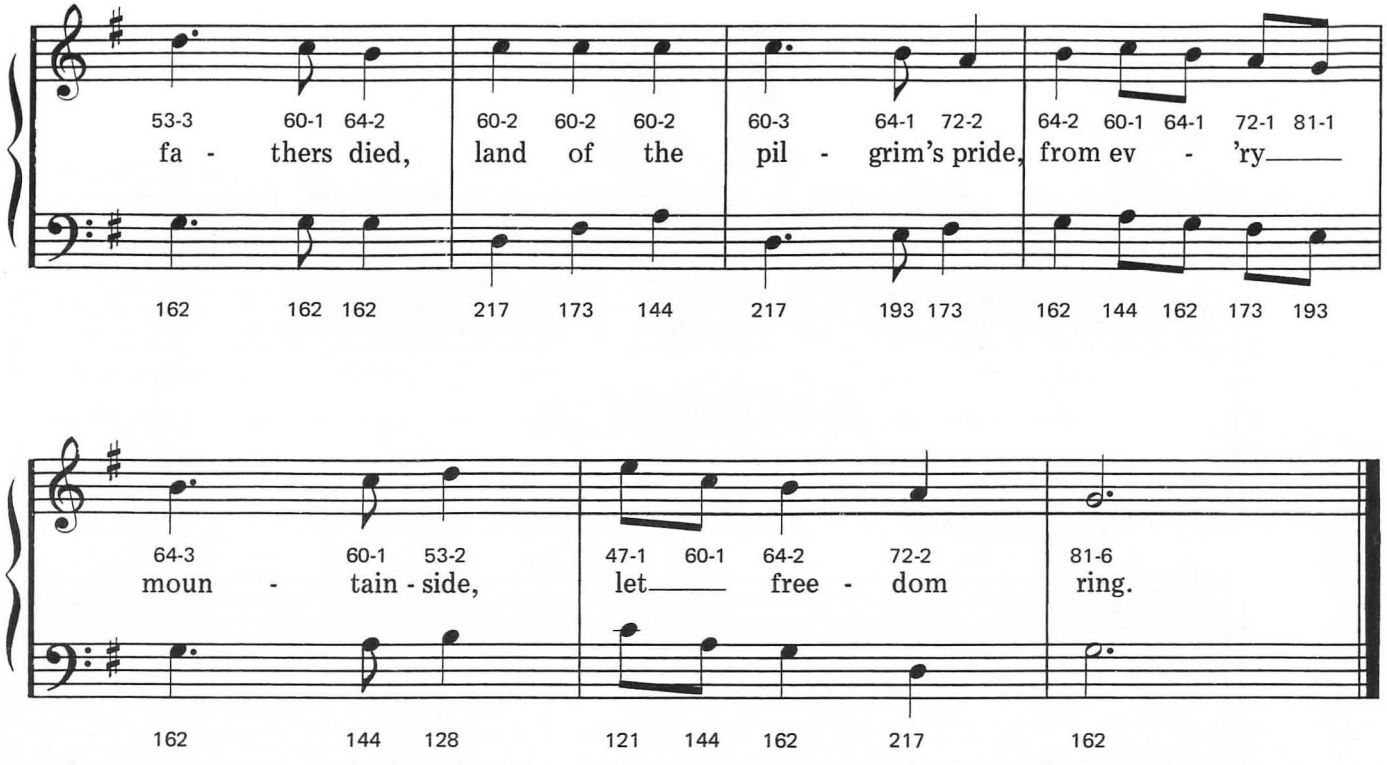
MUSIC ARRANGED FOR THE TWO VOICE PLAYER

AMERICA

Samuel Francis Smith 1831 *Traditional European*

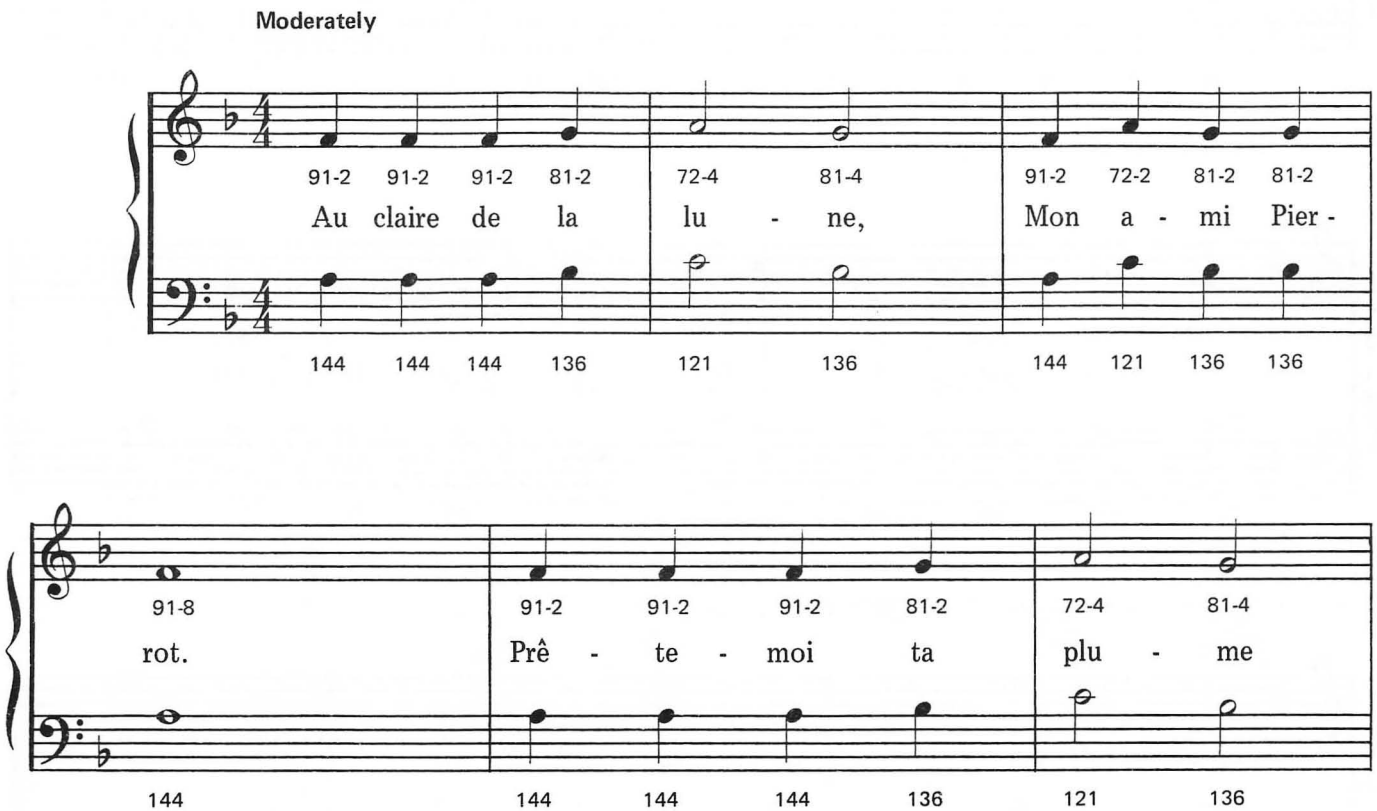
Moderately Slow

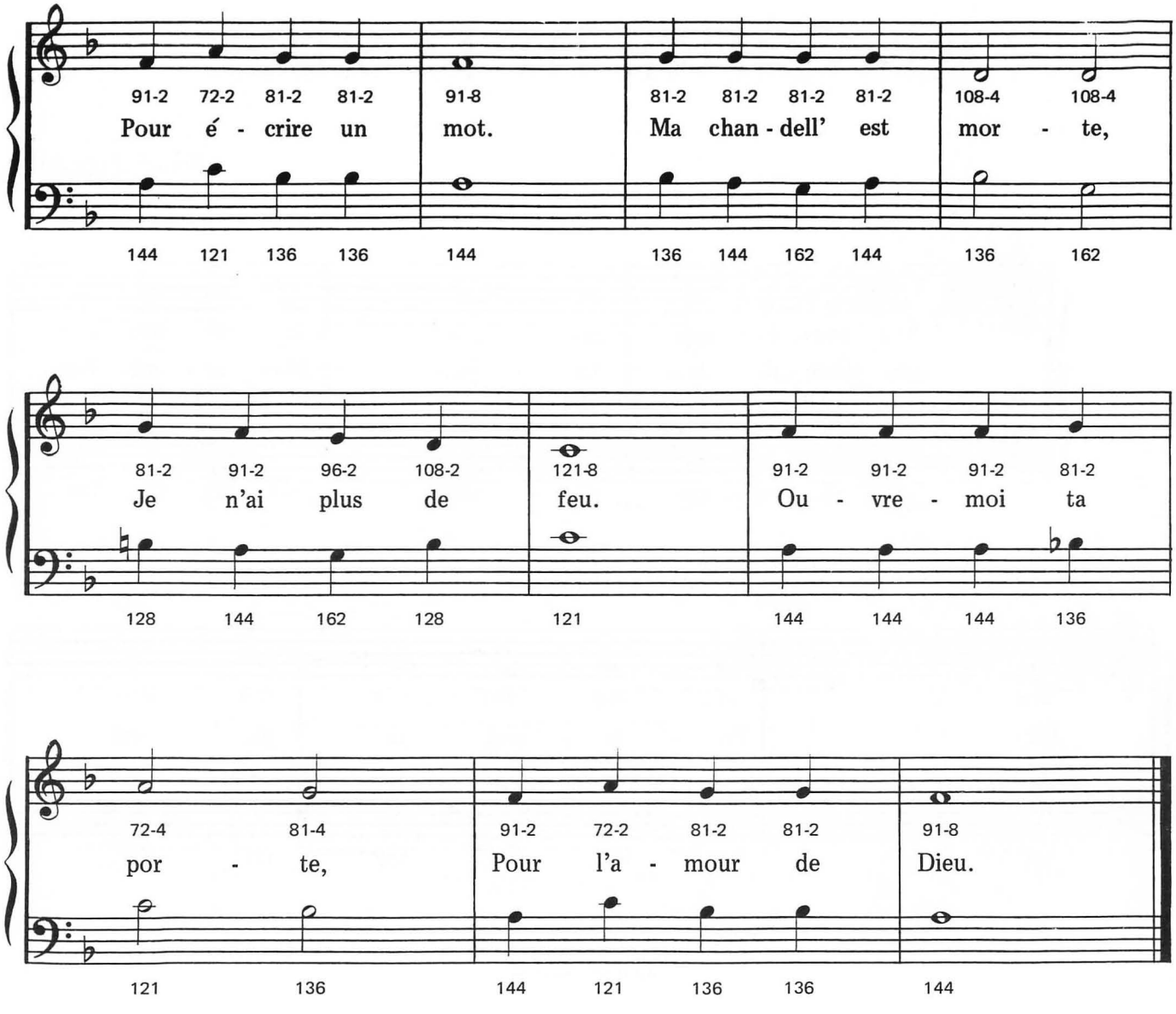



53-3 60-1 64-2 60-2 60-2 60-2 60-3 64-1 72-2 64-2 60-1 64-1 72-1 81-1
fa - thers died, land of the pil - grim's pride, from ev - 'ry
162 162 162 217 173 144 217 193 173 162 144 162 173 193
64-3 60-1 53-2 47-1 60-1 64-2 72-2 81-6
moun - tain - side, let free - dom ring.
162 144 128 121 144 162 217 162

AU CLAIRE DE LA LUNE

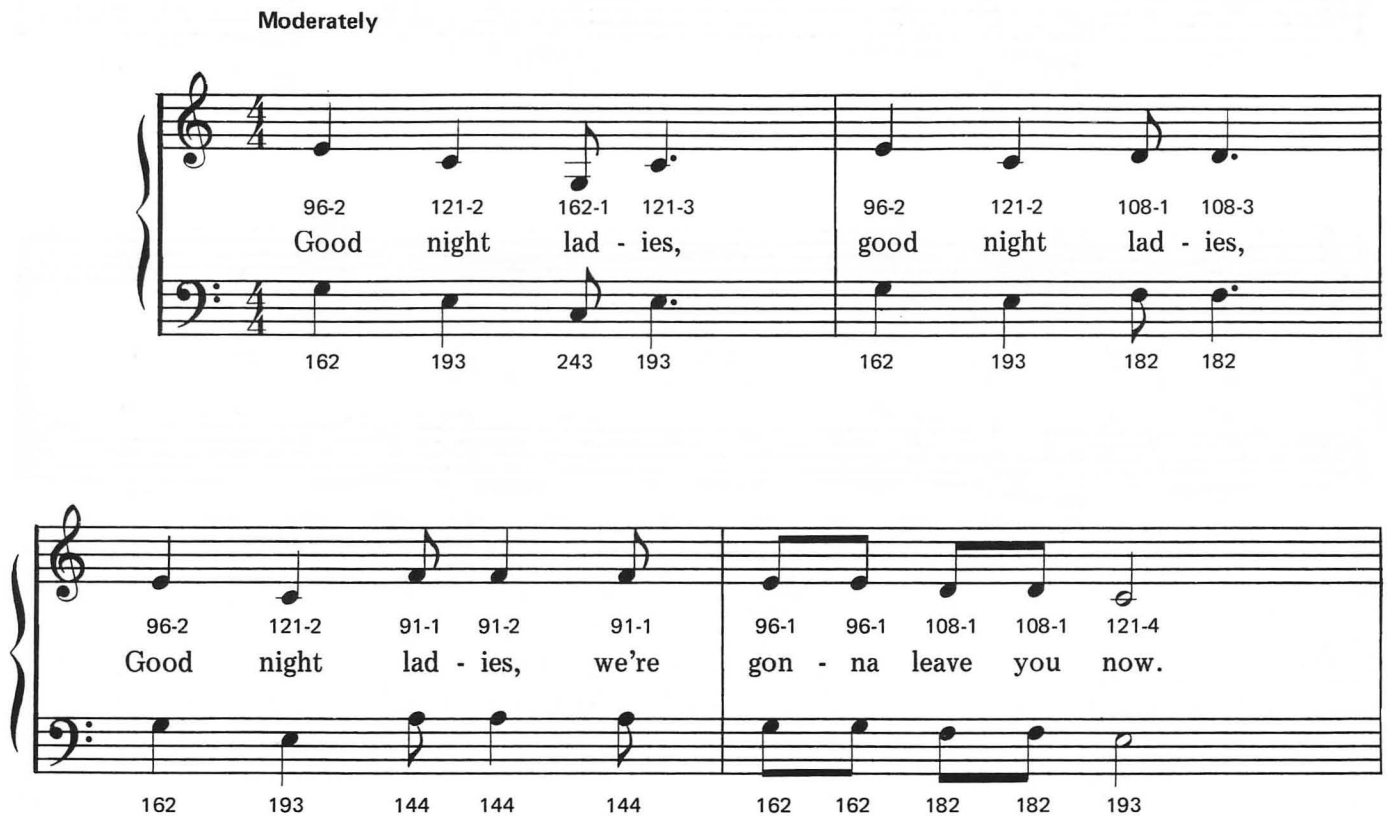
French Folk Song

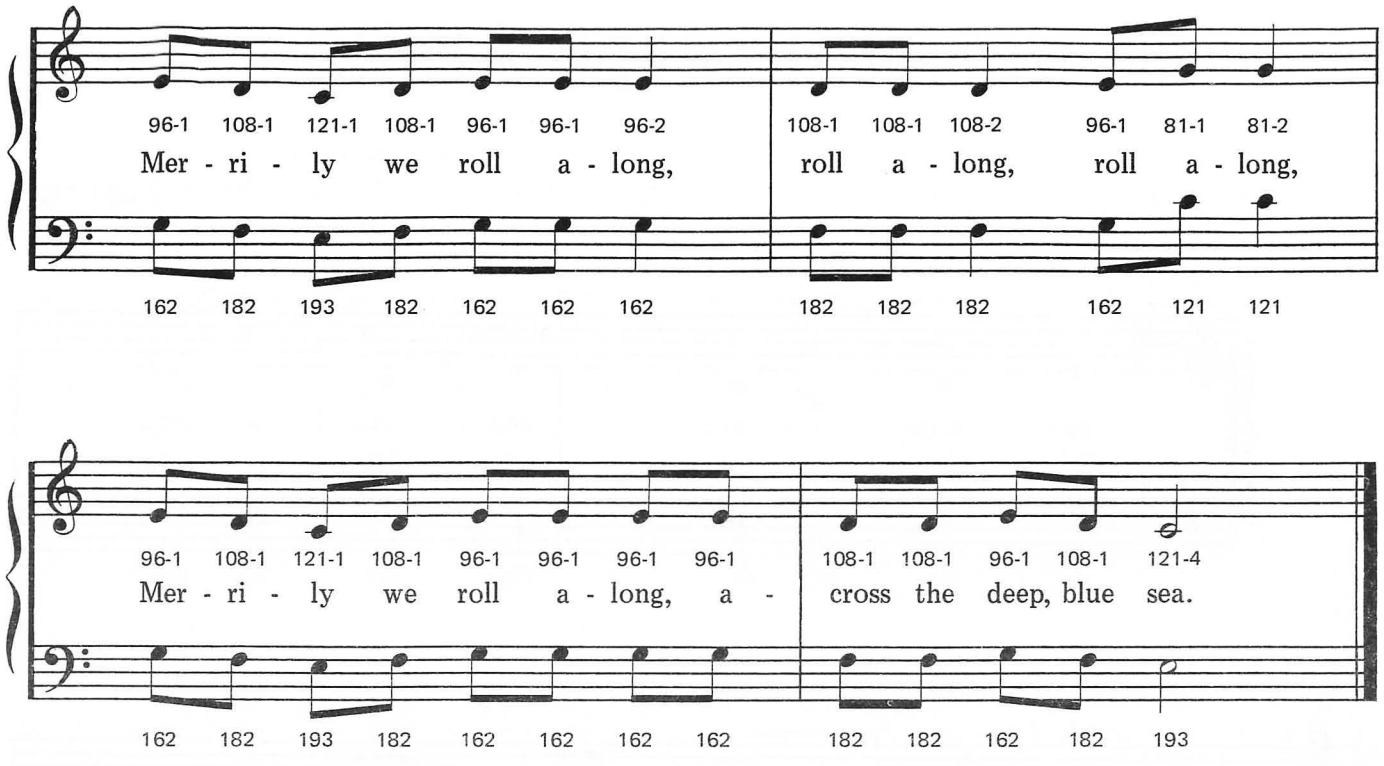



91-2 72-2 81-2 81-2 91-8 81-2 81-2 81-2 81-2 108-4 108-4
Pour é - crire un mot. Ma chan - dell' est mor - te,
144 121 136 136 144 136 144 162 144 136 162
81-2 91-2 96-2 108-2 121-8 91-2 91-2 91-2 81-2
Je n'ai plus de feu. Ou - vre - moi ta
128 144 162 128 121 144 144 144 136
72-4 81-4 91-2 72-2 81-2 81-2 91-8
por - te, Pour l'a - mour de Dieu.
121 136 144 121 136 136 144

GOOD NIGHT LADIES

Traditional

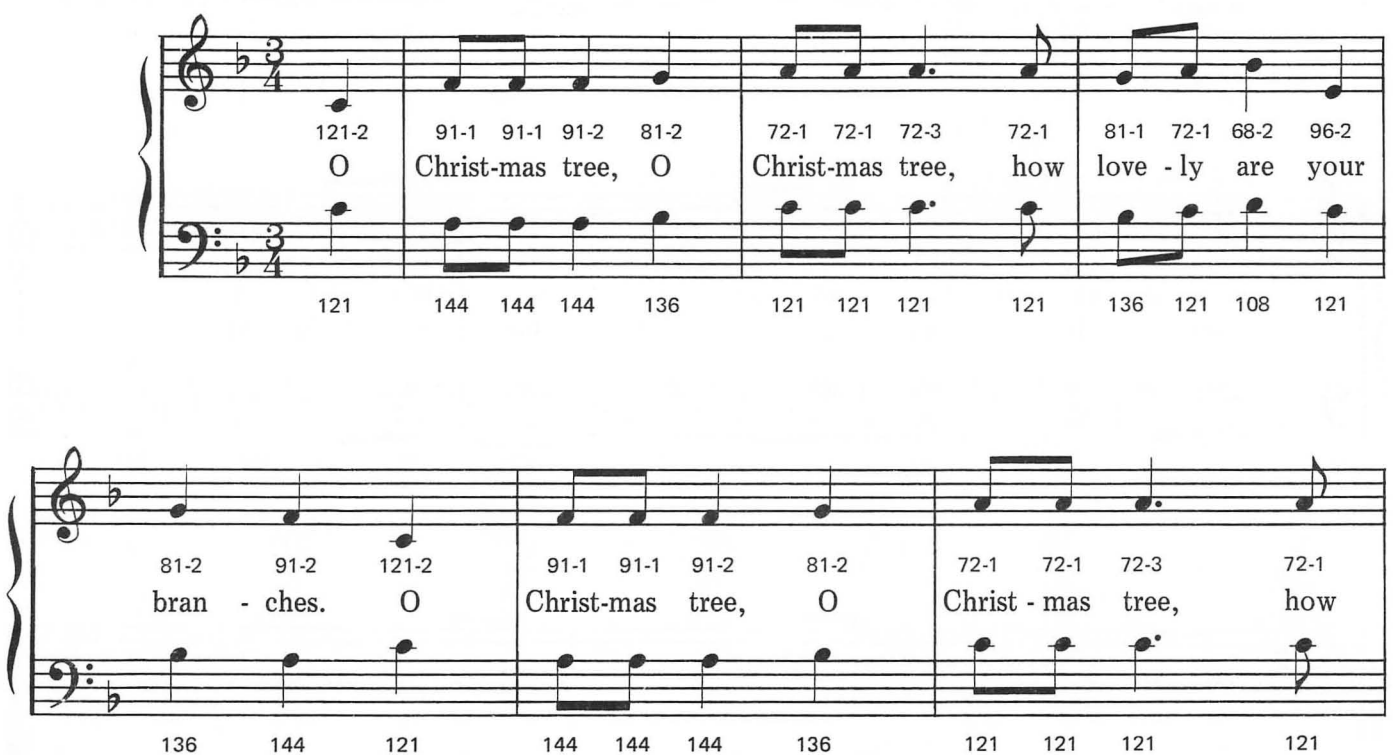


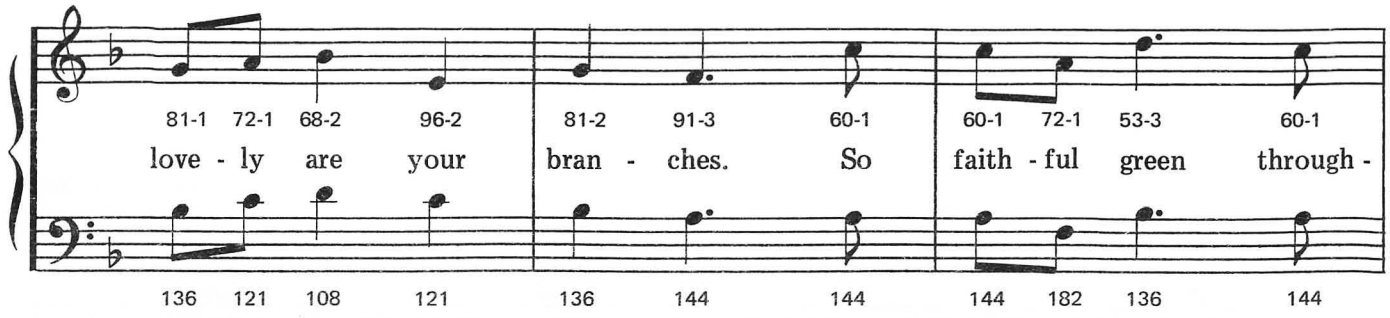

96-1 108-1 121-1 108-1 96-1 96-1 96-2
Mer - ri - ly we roll a - long,
108-1 108-1 108-2 96-1 81-1 81-2
roll a - long, roll a - long,
162 182 193 182 162 162 162
182 182 182 162 121 121
96-1 108-1 121-1 108-1 96-1 96-1 96-1 96-1
Mer - ri - ly we roll a - long, a -
108-1 108-1 96-1 108-1 121-4
cross the deep, blue sea.
162 182 193 182 162 162 162 162
182 182 162 182 193

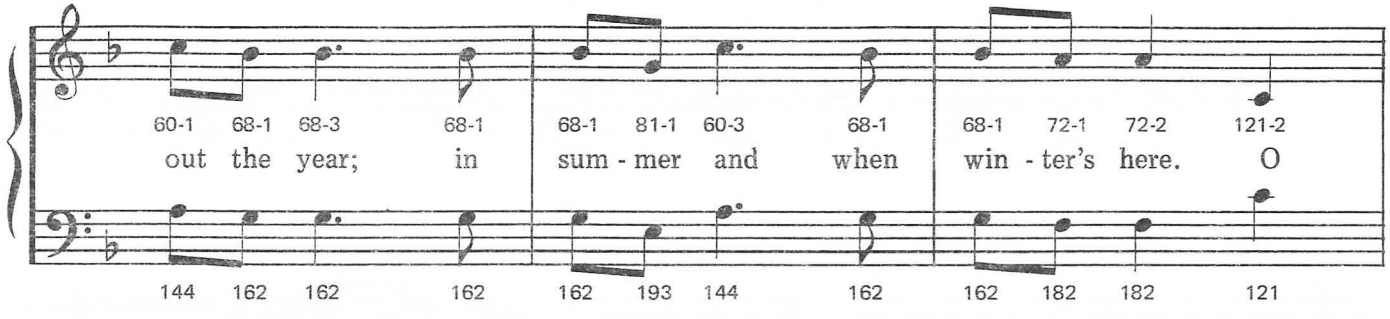
O CHRISTMAS TREE

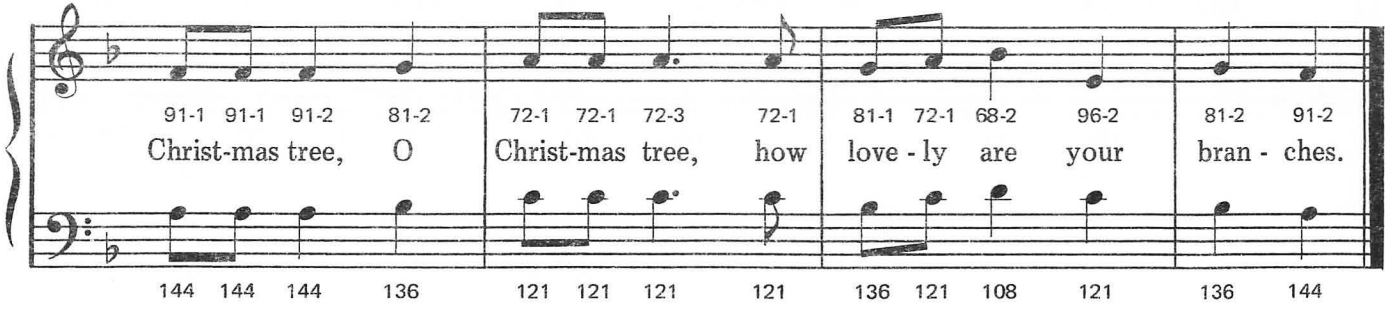
German Folk Song

Moderately




81-1 72-1 68-2 96-2 81-2 91-3 60-1 60-1 72-1 53-3 60-1
love - ly are your bran - ches. So faith - ful green through -
136 121 108 121 136 144 144 144 182 136 144

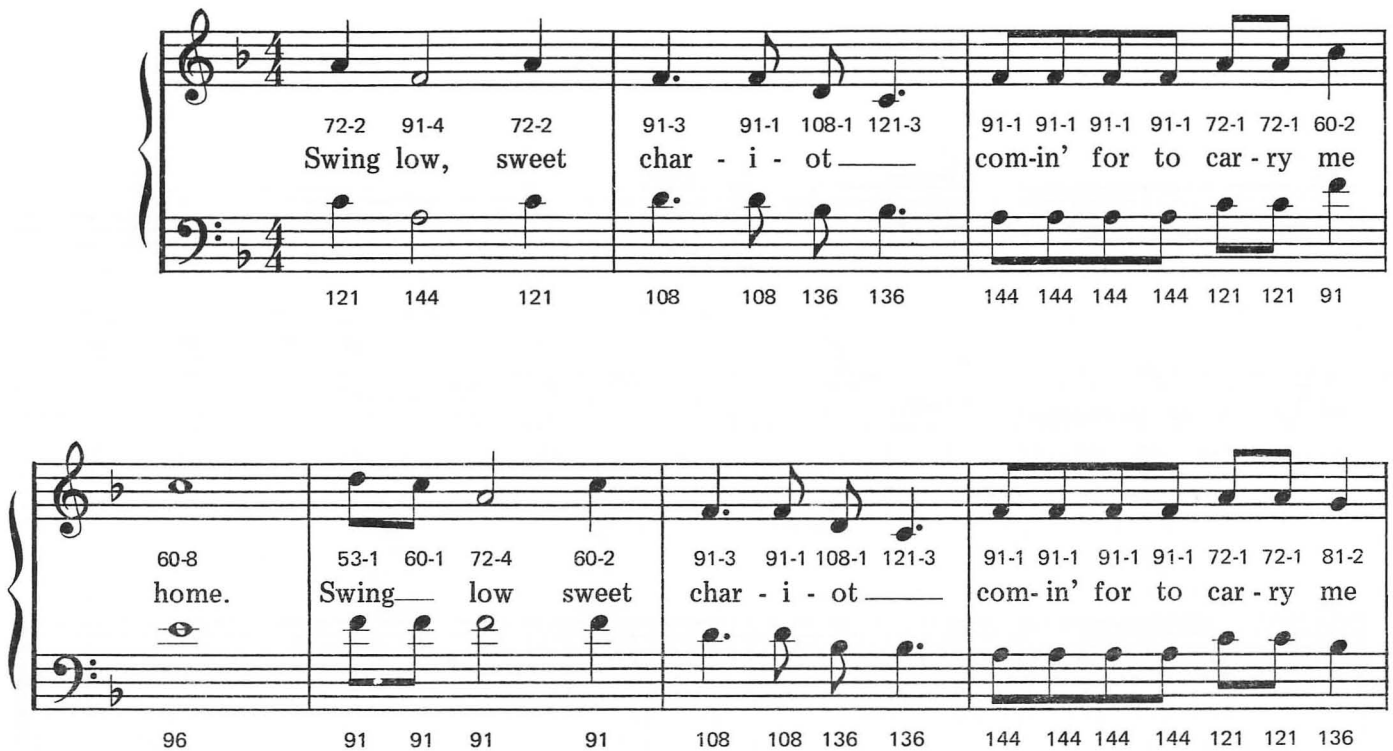

60-1 68-1 68-3 68-1 68-1 81-1 60-3 68-1 68-1 72-1 72-2 121-2
out the year; in sum - mer and when win - ter's here. O
144 162 162 162 162 193 144 162 162 182 182 121

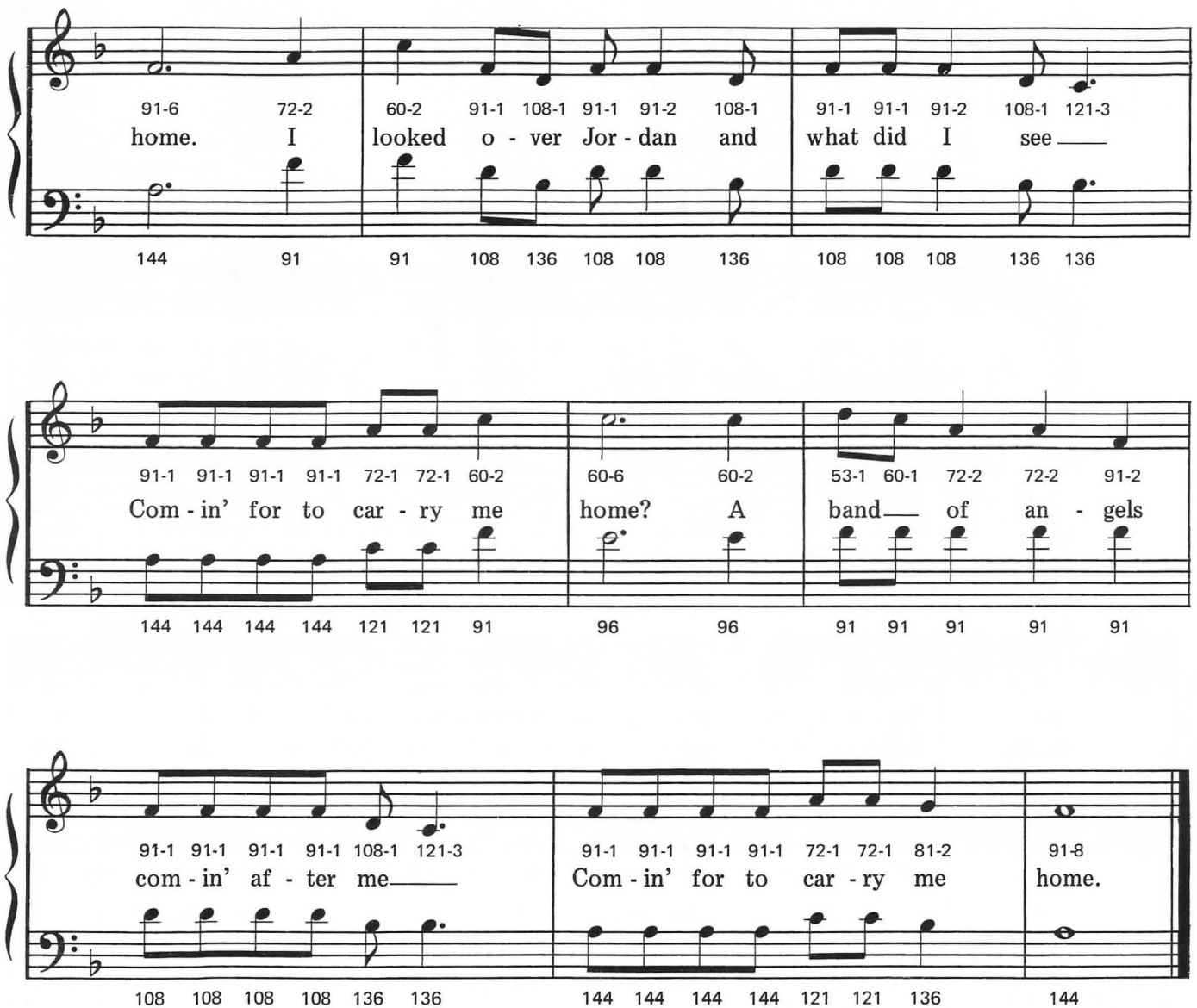

91-1 91-1 91-2 81-2 72-1 72-1 72-3 72-1 81-1 72-1 68-2 96-2 81-2 91-2
Christ-mas tree, O Christ-mas tree, how love - ly are your bran - ches.
144 144 144 136 121 121 121 121 136 121 108 121 136 144

SWING LOW, SWEET CHARIOT

American Spiritual

Moderately Slow



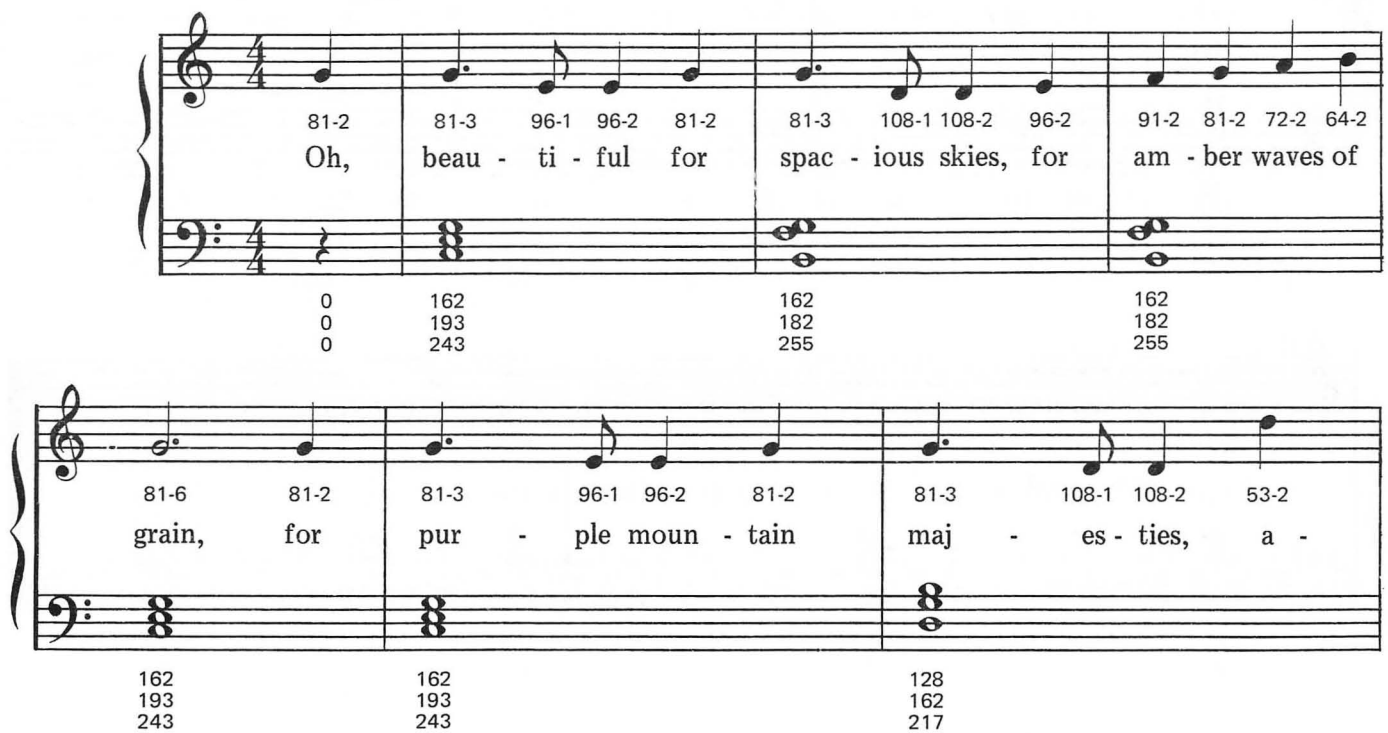

91-6 72-2 60-2 91-1 108-1 91-1 91-2 108-1 91-1 91-1 91-2 108-1 121-3
home. I looked o - ver Jor - dan and what did I see
144 91 91 108 136 108 108 136 108 108 108 136 136
91-1 91-1 91-1 91-1 72-1 72-1 60-2 60-6 60-2 53-1 60-1 72-2 72-2 91-2
Com - in' for to car - ry me home? A band of an - gels
144 144 144 144 121 121 91 96 96 91 91 91 91 91
91-1 91-1 91-1 91-1 108-1 121-3 91-1 91-1 91-1 91-1 72-1 72-1 81-2 91-8
com - in' af - ter me Com - in' for to car - ry me home.
108 108 108 108 136 136 144 144 144 144 121 121 136 144

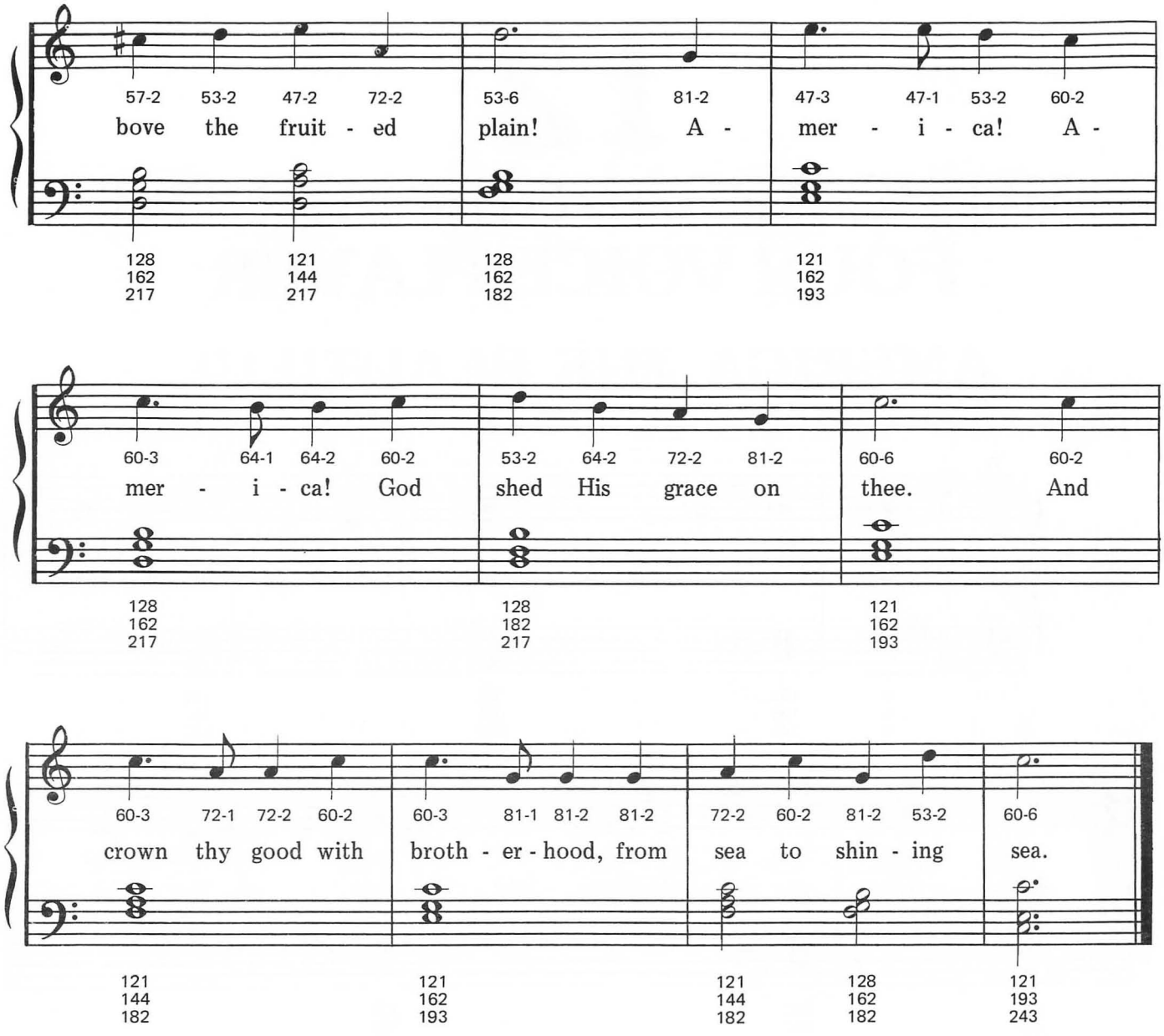
12

MUSIC ARRANGED FOR THE FOUR VOICE PLAYER

AMERICA THE BEAUTIFUL

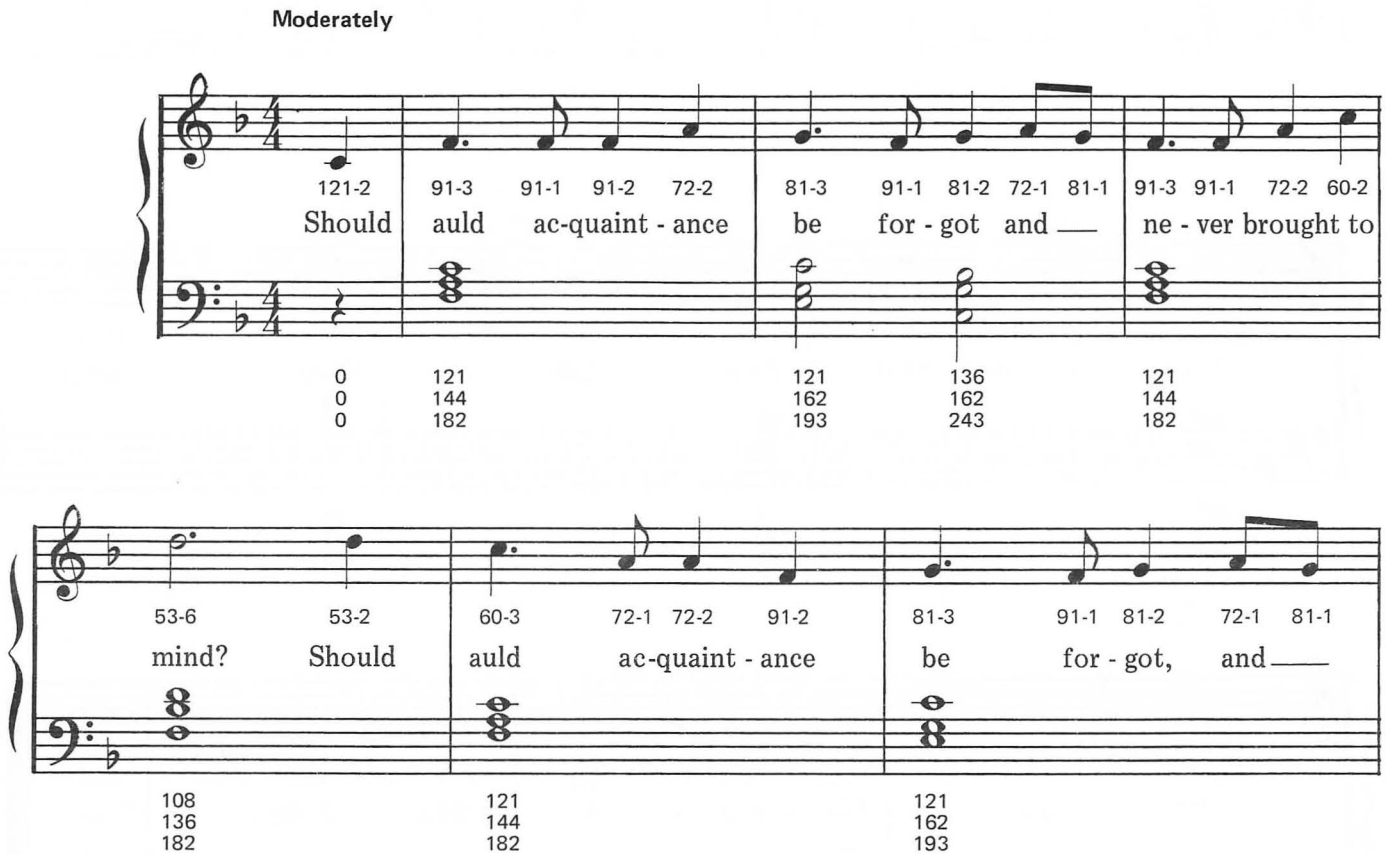
Katherine L. Bates *Samuel A. Ward*

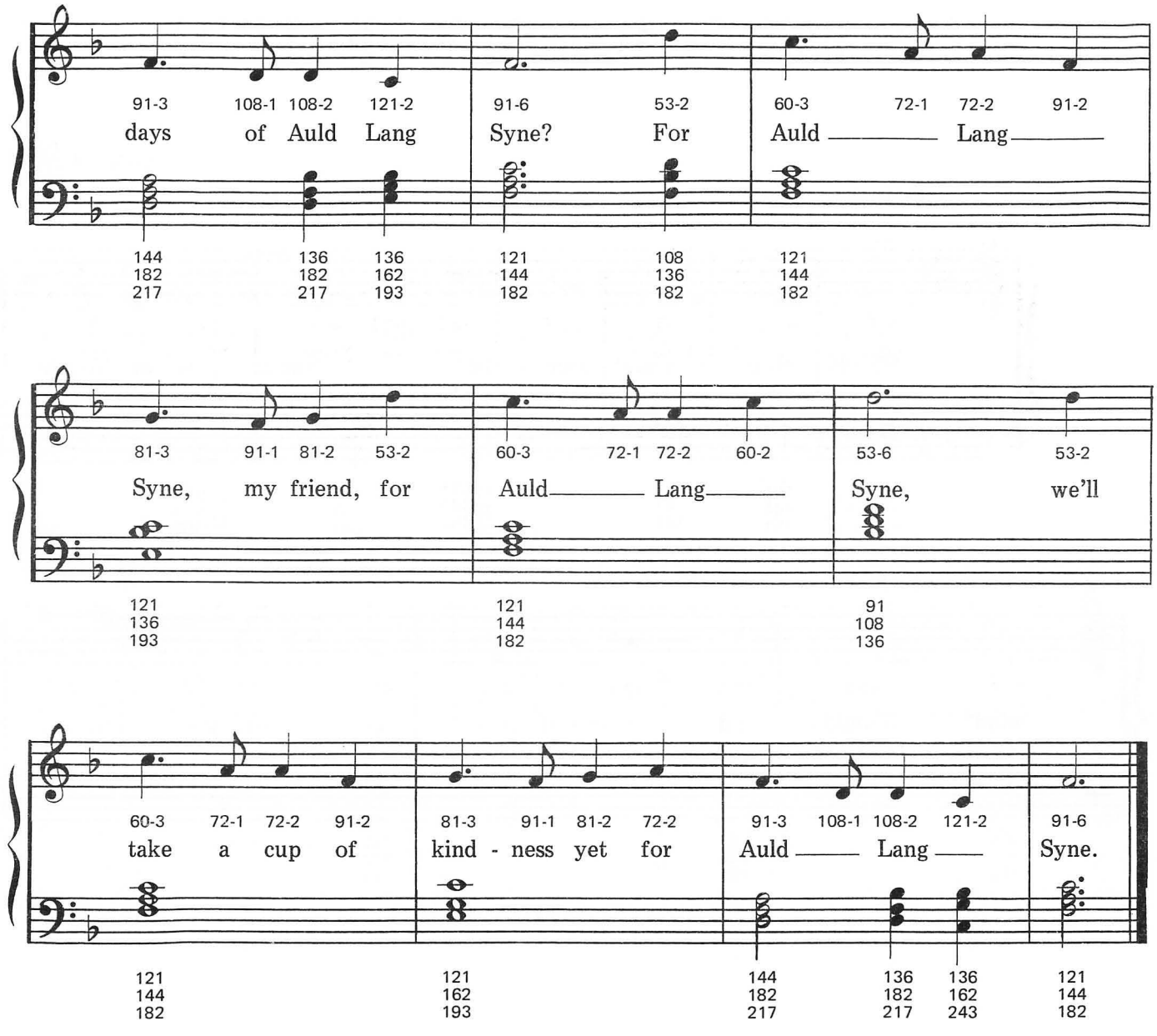



57-2 53-2 47-2 72-2 53-6 81-2 47-3 47-1 53-2 60-2
bove the fruit - ed plain! A - mer - i - ca! A -
128 162 217
121 144 217
128 162 182
121 162 193
60-3 64-1 64-2 60-2 53-2 64-2 72-2 81-2 60-6 60-2
mer - i - ca! God shed His grace on thee. And
128 162 217
128 182 217
121 162 193
60-3 72-1 72-2 60-2 60-3 81-1 81-2 81-2 72-2 60-2 81-2 53-2 60-6
crown thy good with broth - er - hood, from sea to shin - ing sea.
121 144 182
121 162 193
121 144 182
128 162 182
121 193 243

AULD LANG SYNE

Traditional Scottish

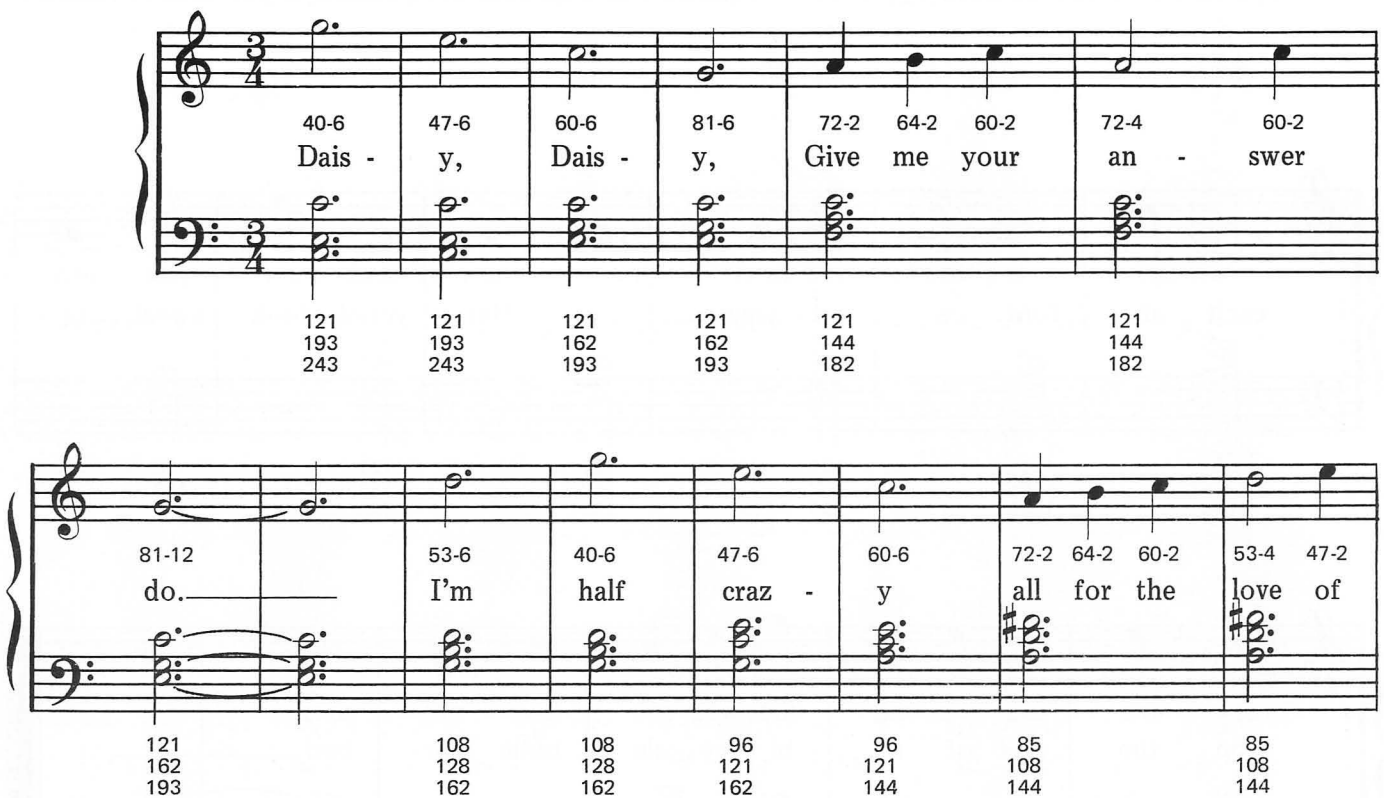


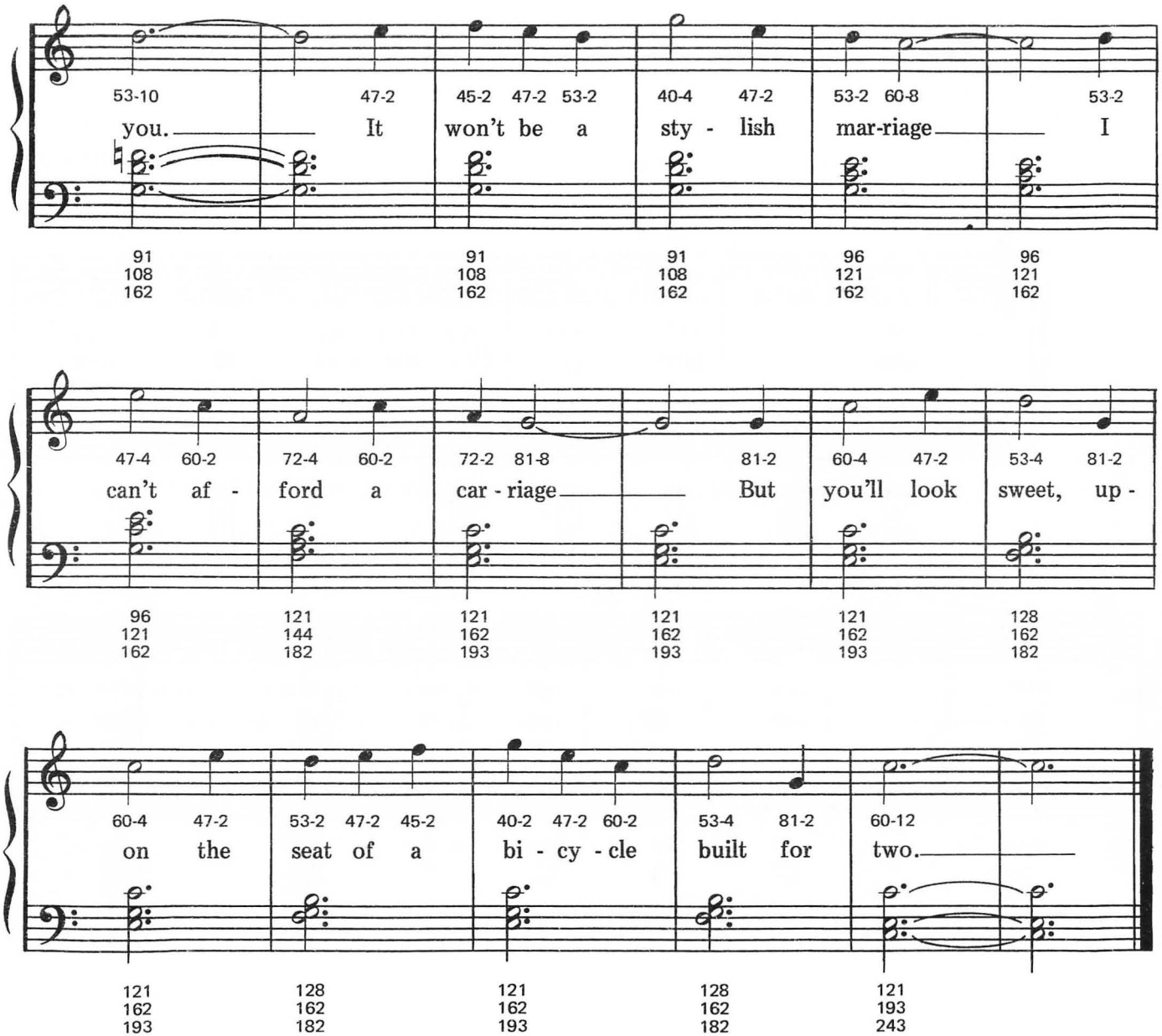

91-3 108-1 108-2 121-2 91-6 53-2 60-3 72-1 72-2 91-2
days of Auld Lang Syne? For Auld ____ Lang ____
144 182 217 136 182 217 136 162 193 121 144 182 108 136 182 121 144 182
81-3 91-1 81-2 53-2 60-3 72-1 72-2 60-2 53-6 53-2
Syne, my friend, for Auld ____ Lang ____ Syne, we'll
121 136 193 121 144 182 91 108 136
60-3 72-1 72-2 91-2 81-3 91-1 81-2 72-2 91-3 108-1 108-2 121-2 91-6
take a cup of kind - ness yet for Auld ____ Lang ____ Syne.
121 144 182 121 162 193 144 182 217 136 182 217 136 162 243 121 144 182

BICYCLE BUILT FOR TWO

Henry Dacre

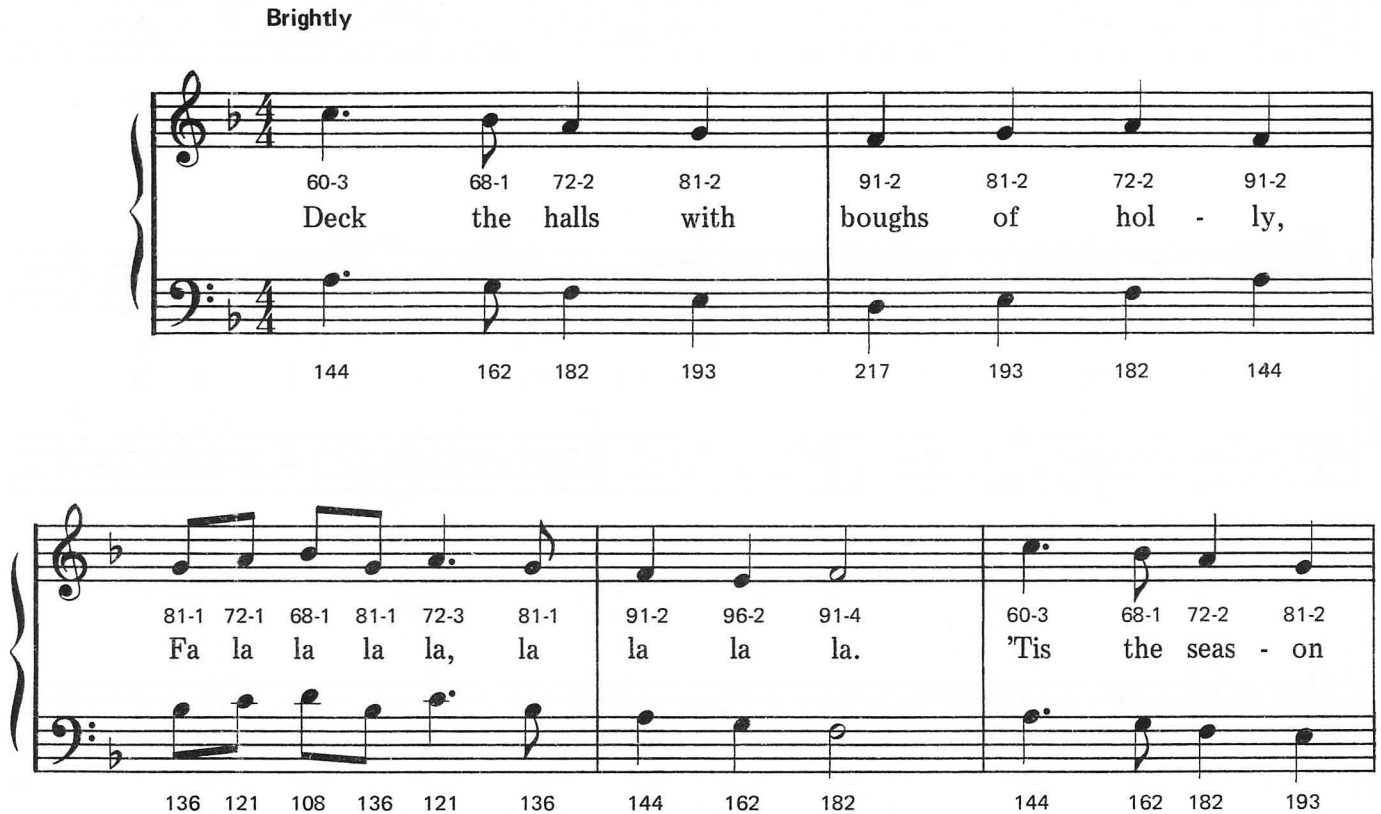
Moderately

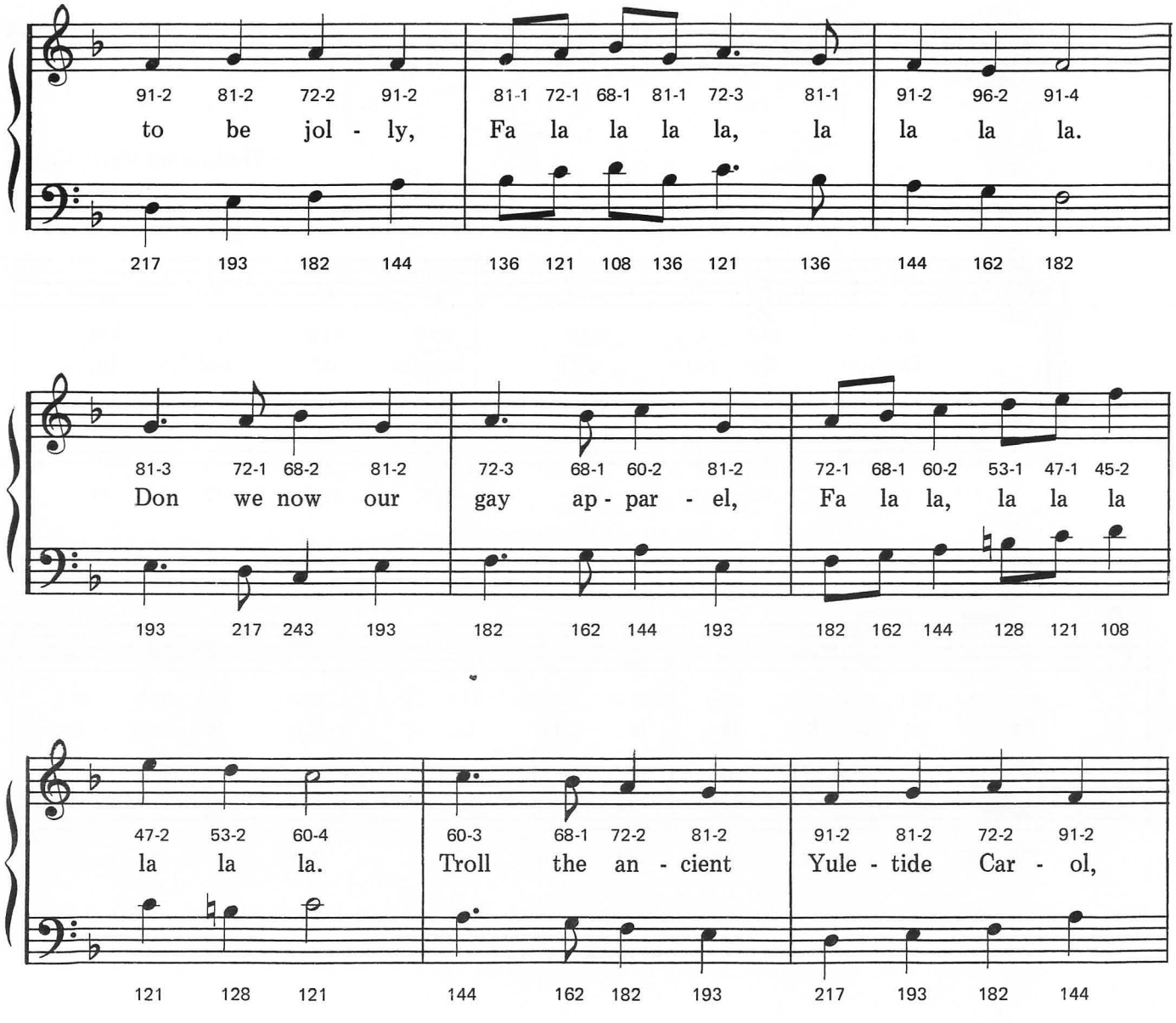


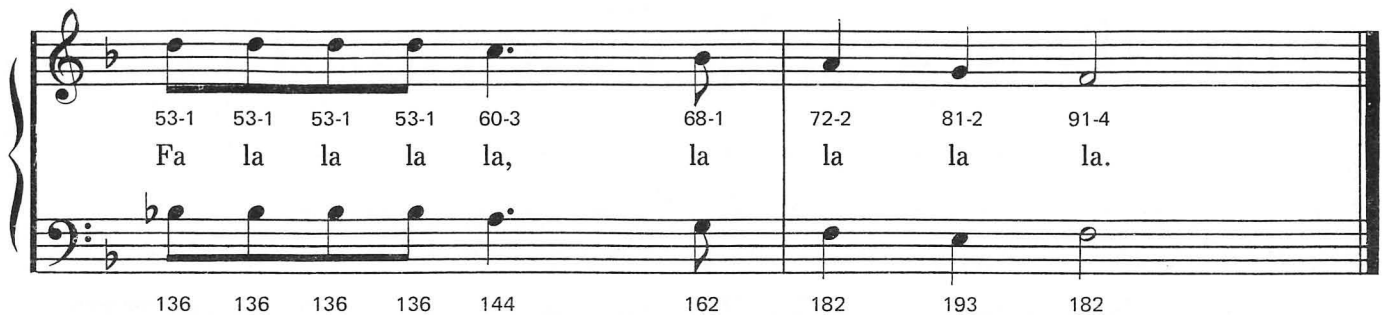

53-10 47-2 45-2 47-2 53-2 40-4 47-2 53-2 60-8 53-2
you. It won't be a sty - lish mar-riage I
91 108 162 | 91 108 162 | 91 108 162 | 96 121 162 | 96 121 162
47-4 60-2 72-4 60-2 72-2 81-8 81-2 60-4 47-2 53-4 81-2
can't af - ford a car - riage But you'll look sweet, up -
96 121 162 | 121 144 182 | 121 162 193 | 121 162 193 | 121 162 193 | 128 162 182
60-4 47-2 53-2 47-2 45-2 40-2 47-2 60-2 53-4 81-2 60-12
on the seat of a bi - cy - cle built for two.
121 162 193 | 128 162 182 | 121 162 193 | 128 162 182 | 121 193 243

DECK THE HALLS

Traditional Welsh Carol




91-2 81-2 72-2 91-2 81-1 72-1 68-1 81-1 72-3 81-1 91-2 96-2 91-4
to be jol - ly, Fa la la la la, la la la la.
217 193 182 144 136 121 108 136 121 136 144 162 182
81-3 72-1 68-2 81-2 72-3 68-1 60-2 81-2 72-1 68-1 60-2 53-1 47-1 45-2
Don we now our gay ap - par - el, Fa la la, la la la
193 217 243 193 182 162 144 193 182 162 144 128 121 108
47-2 53-2 60-4 60-3 68-1 72-2 81-2 91-2 81-2 72-2 91-2
la la la. Troll the an - cient Yule - tide Car - ol,
121 128 121 144 162 182 193 217 193 182 144

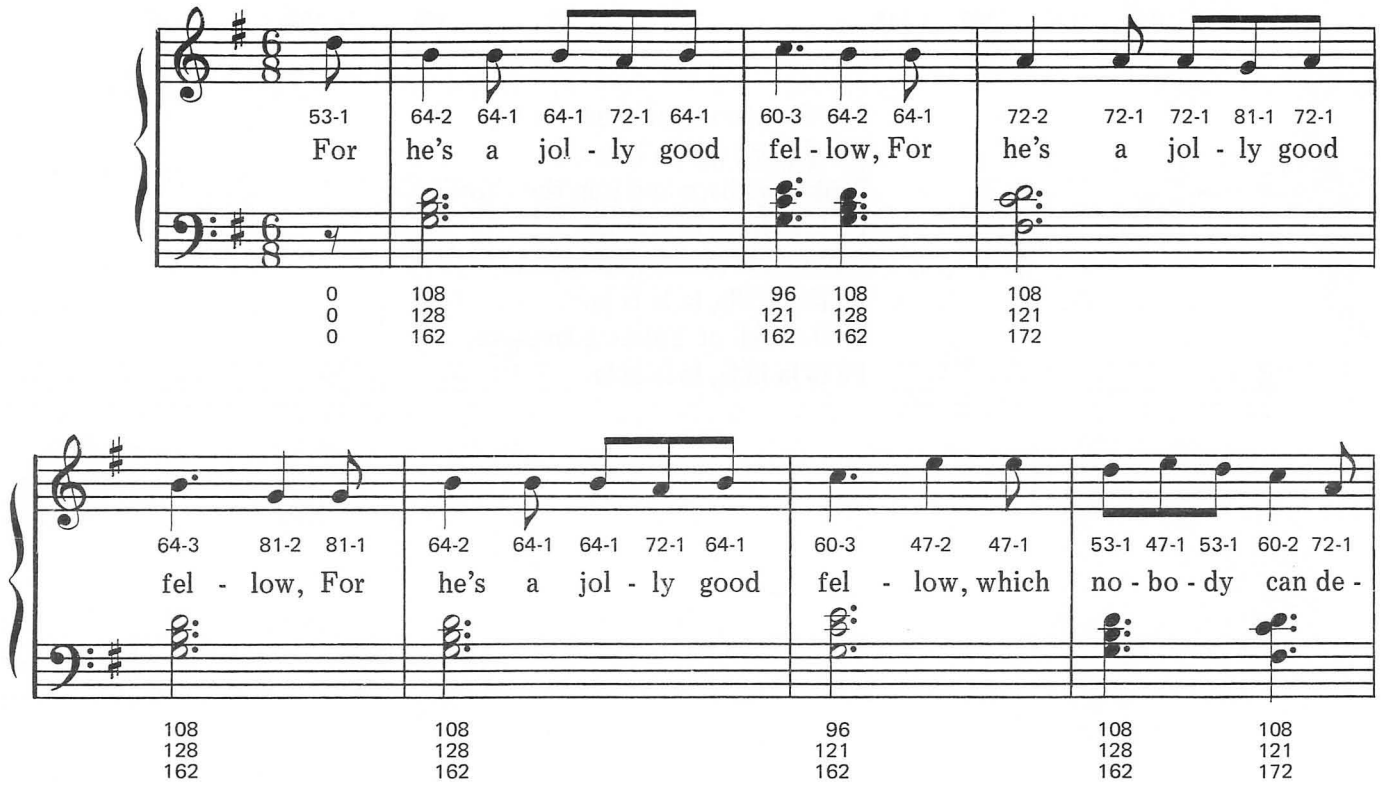


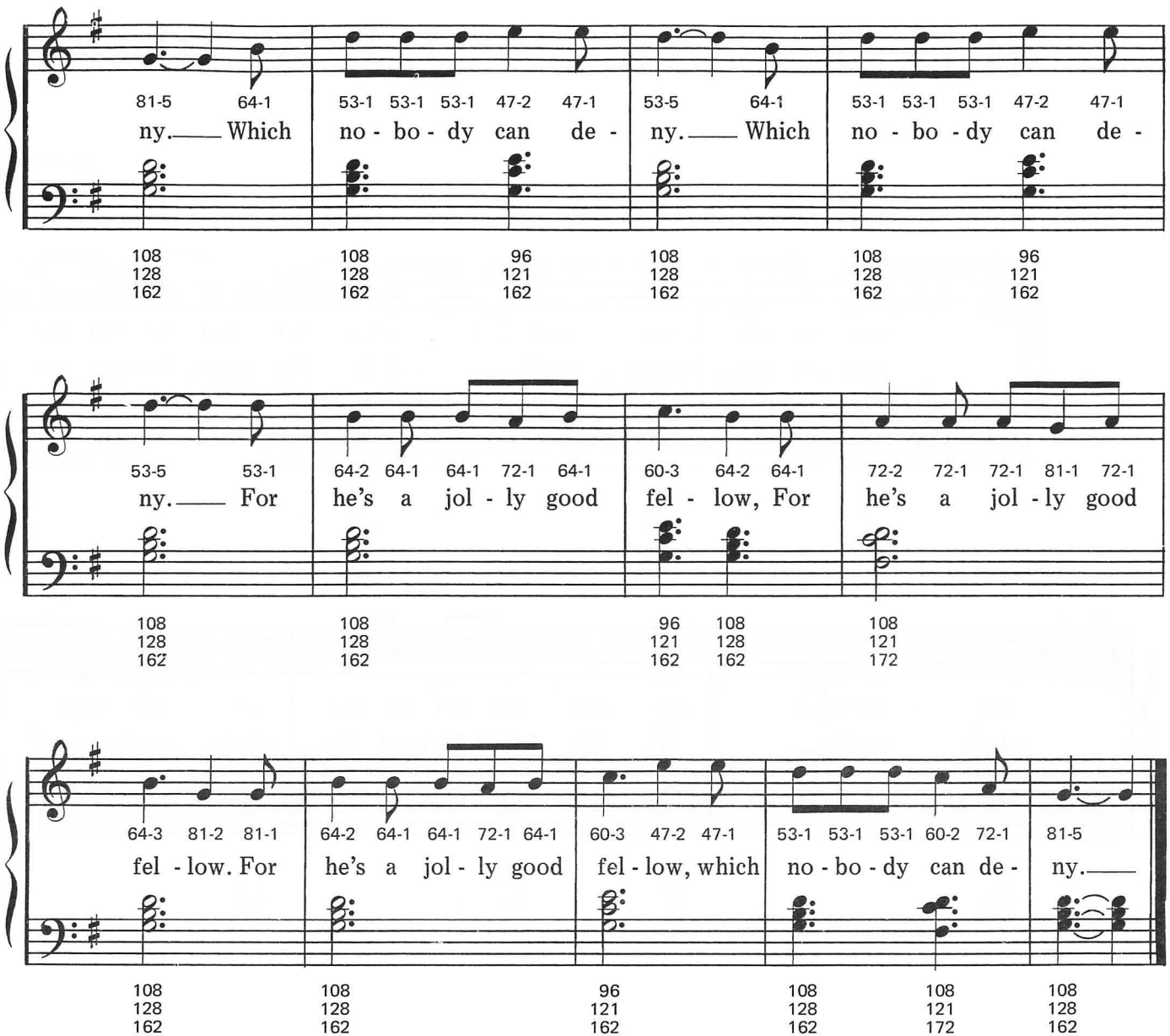
2. See the blazing yule before us,
Fa la la la la, la la la la.
Strike the harp and join the chorus,
Fa la la la la, la la la la.
Follow me in merry measure,
Fa la la la la, la la la la.
While I tell of Yuletide treasure,
Fa la la la la, la la la la.

FOR HE'S A JOLLY GOOD FELLOW

Traditional

Moderately

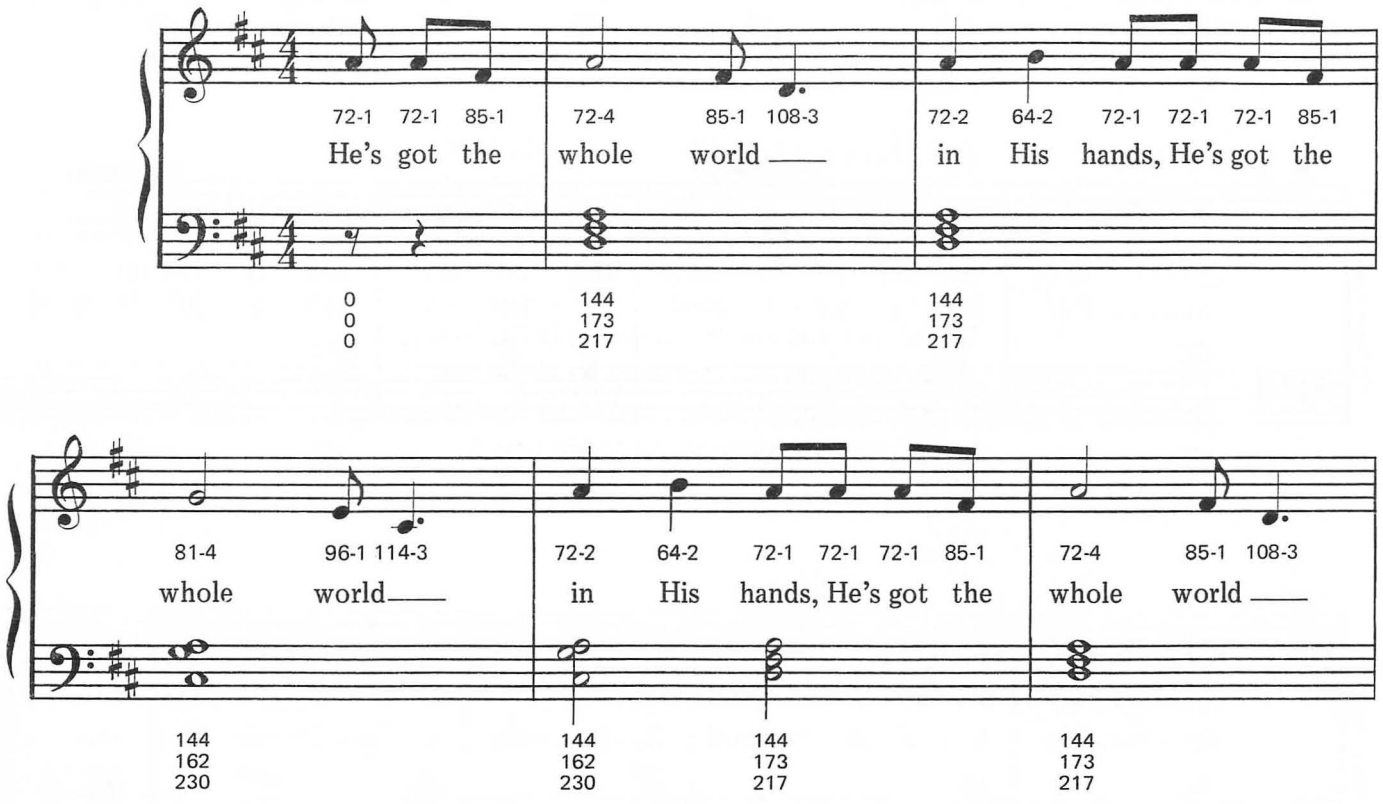


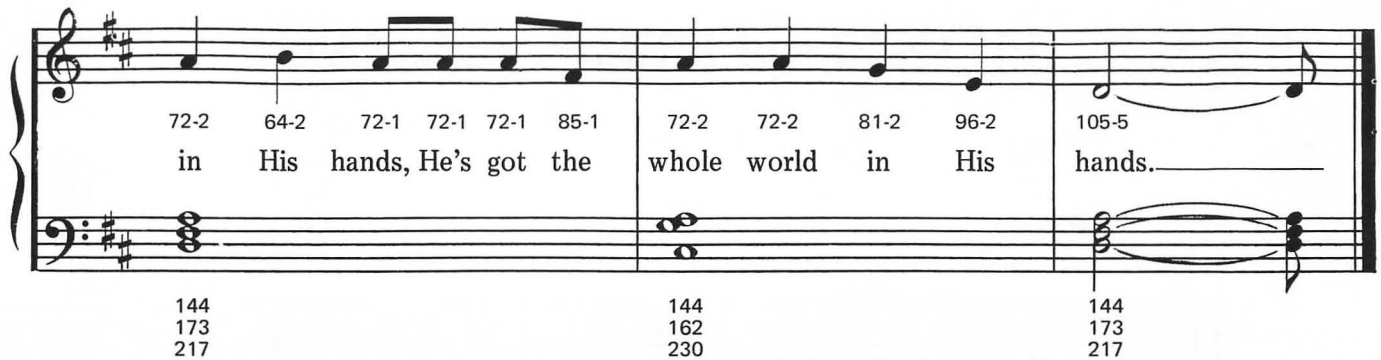

81-5 64-1 53-1 53-1 53-1 47-2 47-1 53-5 64-1 53-1 53-1 53-1 47-2 47-1
ny. Which no - bo - dy can de - ny. Which no - bo - dy can de -
108 128 162 108 128 162 96 121 162 108 128 162 108 128 162 96 121 162
53-5 53-1 64-2 64-1 64-1 72-1 64-1 60-3 64-2 64-1 72-2 72-1 72-1 81-1 72-1
ny. For he's a jol - ly good fel - low, For he's a jol - ly good
108 128 162 108 128 162 96 121 162 108 128 162 108 121 172
64-3 81-2 81-1 64-2 64-1 64-1 72-1 64-1 60-3 47-2 47-1 53-1 53-1 53-1 60-2 72-1 81-5
fel - low. For he's a jol - ly good fel - low, which no - bo - dy can de - ny.
108 128 162 108 128 162 96 121 162 108 128 162 108 121 172 108 128 162

HE'S GOT THE WHOLE WORLD

Spiritual

Moderately





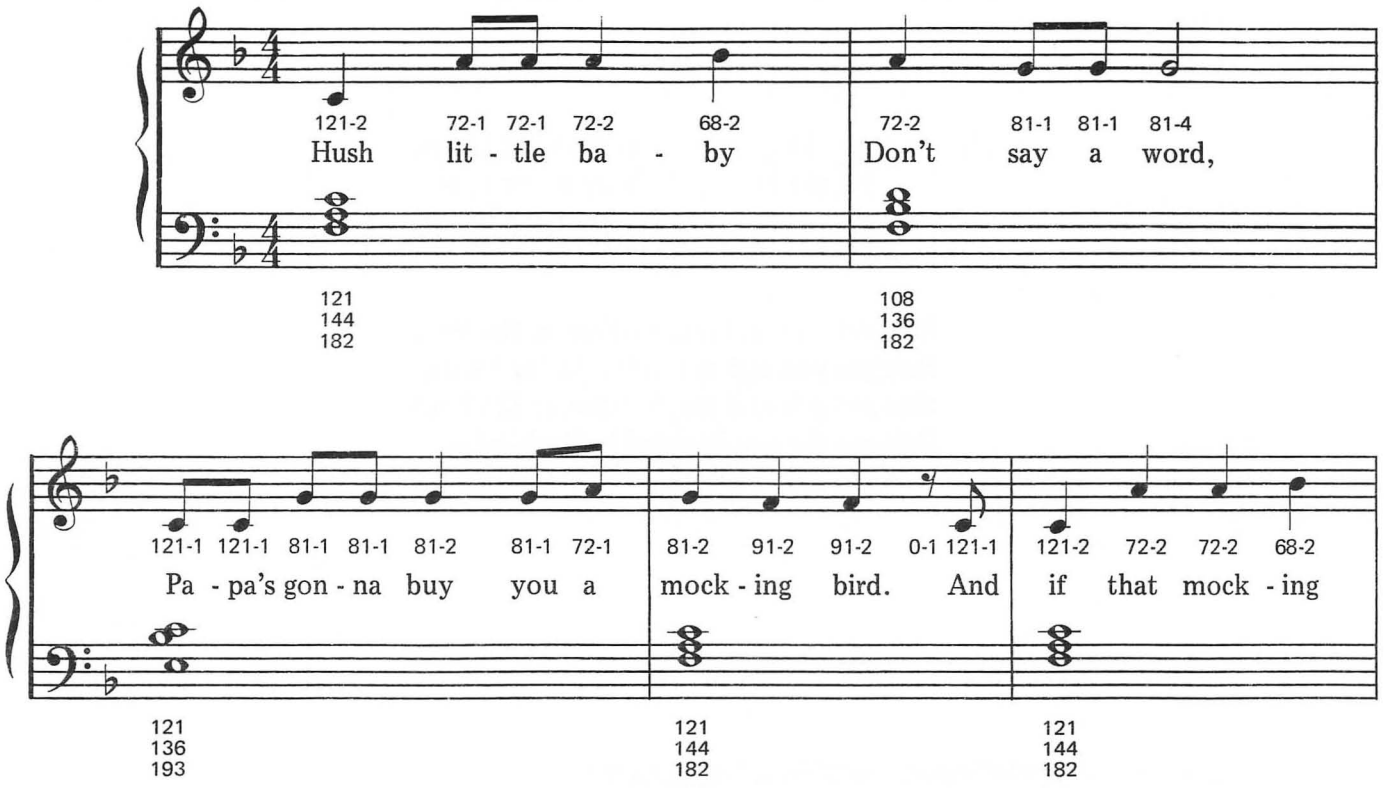
2. He's got the little bitty baby in His hands,
   He's got the little bitty baby in His hands,
   He's got the little bitty baby in His hands,
   He's got the whole world in His hands.

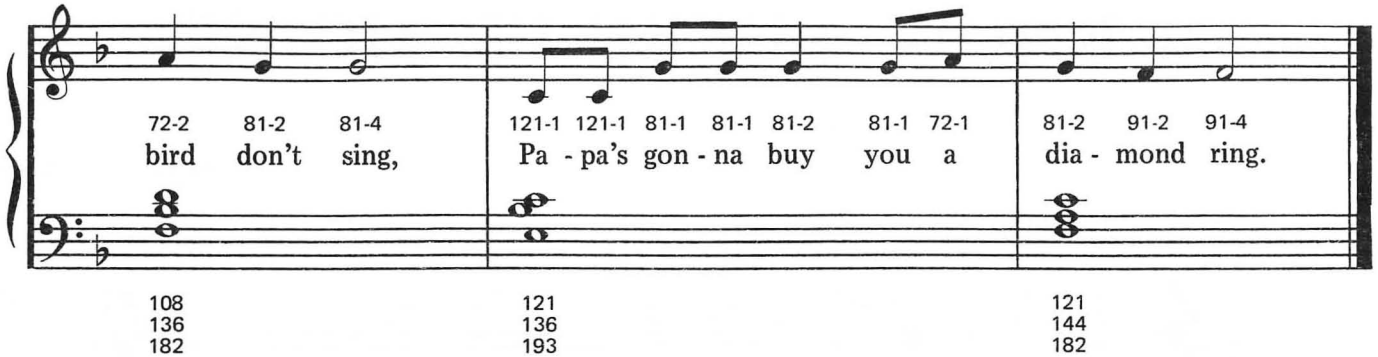
3. He's got you and me, brother, in His hands,
   He's got you and me, sister, in His hands,
   He's got you and me, brother, in His hands,
   He's got the whole world in His hands.

HUSH LITTLE BABY

Lullaby

Moderately





2. And if that diamond ring turns brass,
Papa's gonna buy you a lookin' glass.
And if that lookin' glass gets broke,
Papa's gonna buy you a billy goat.

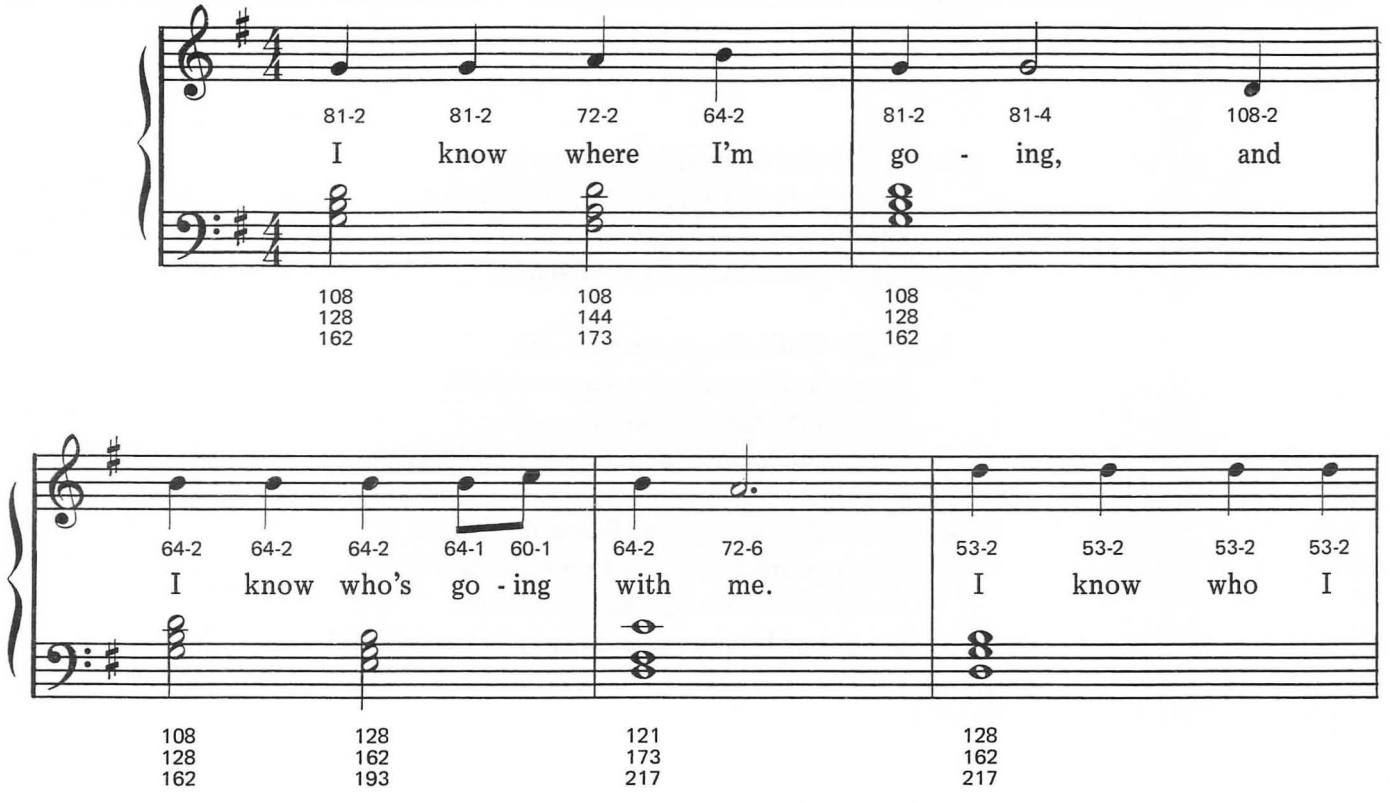
3. And if that billy goat won't pull,
Papa's gonna buy you a cart and bull.
And if that cart and bull turn over,
Papa's gonna buy you a dog named Rover.

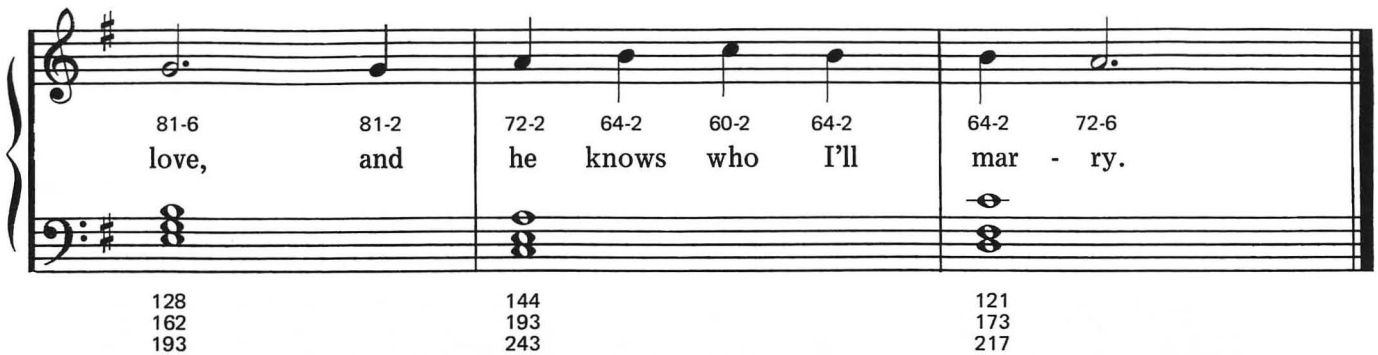
4. And if that dog named Rover don't bark,
Papa's gonna buy you a horse and cart.
And if that horse and cart fall down,
You'll still be the sweetest baby in the town.

I KNOW WHERE I'M GOING

British Folk Song

Moderately Slow





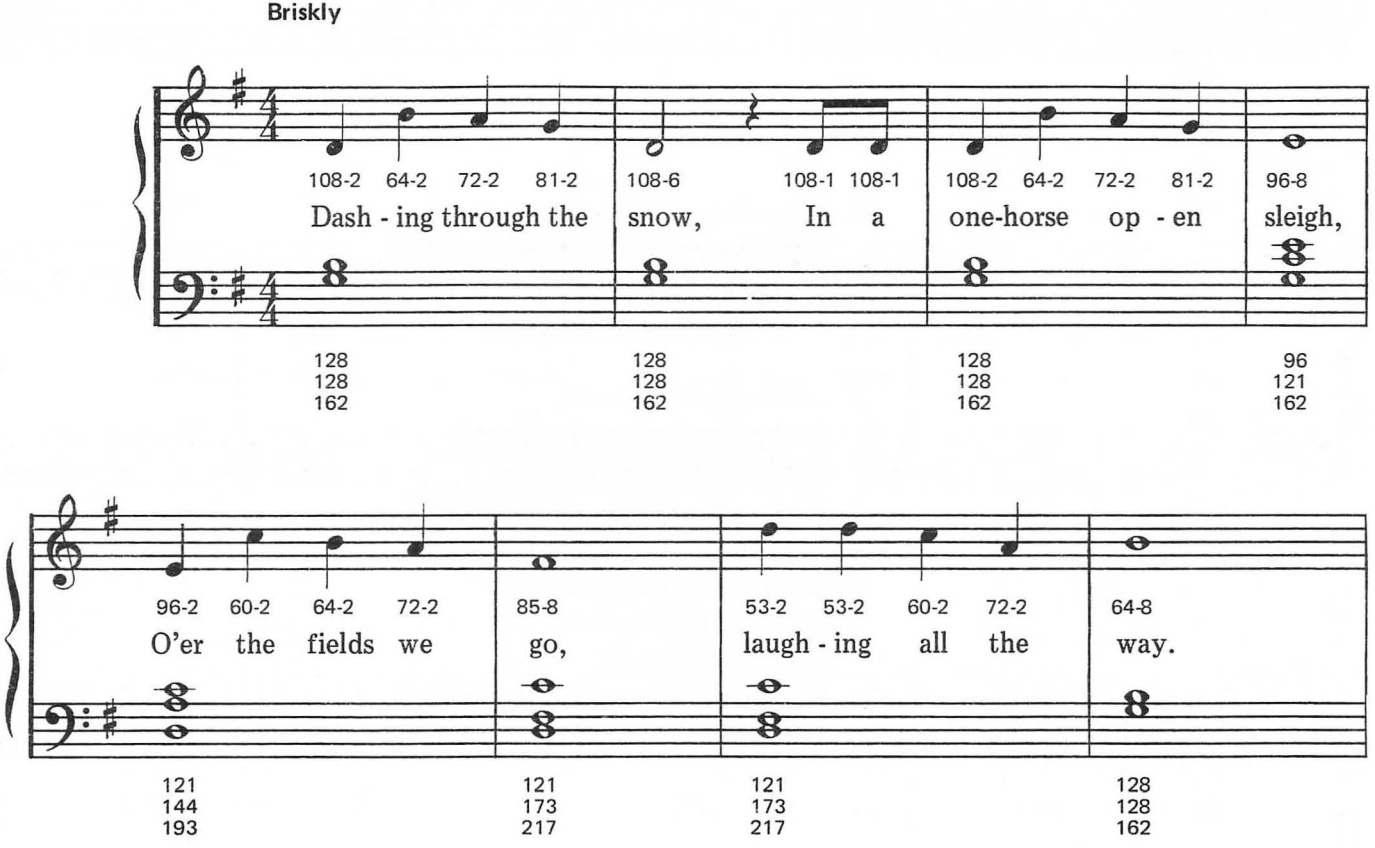
2. I'll give up silk stocking
   And shoes of bright green leather,
   Combs to buckle my hair,
   And rings for every finger.

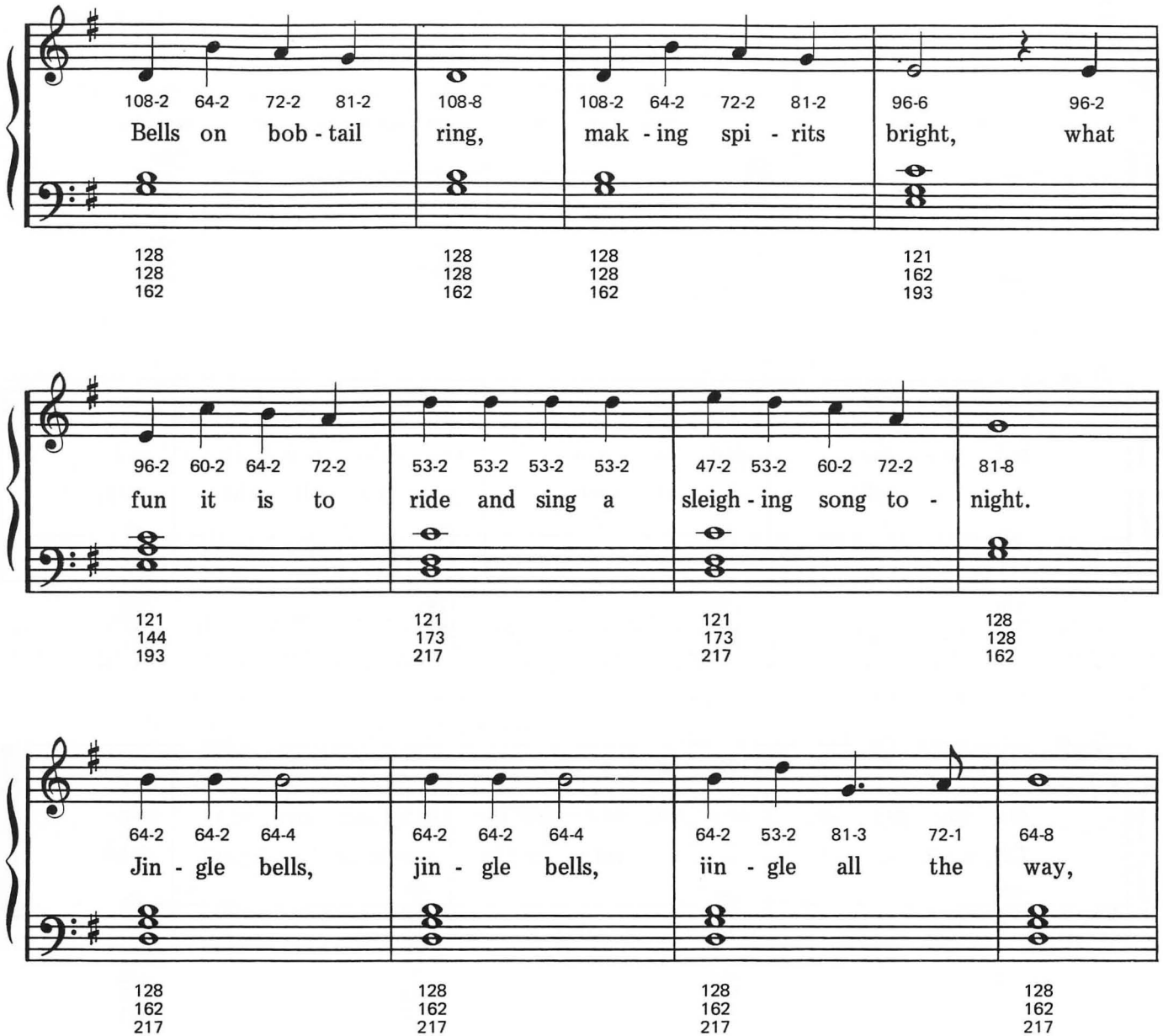
3. Feather beds are soft,
   And painted rooms are bonnie,
   But I would trade them all,
   For my handsome, winsome Johnny.

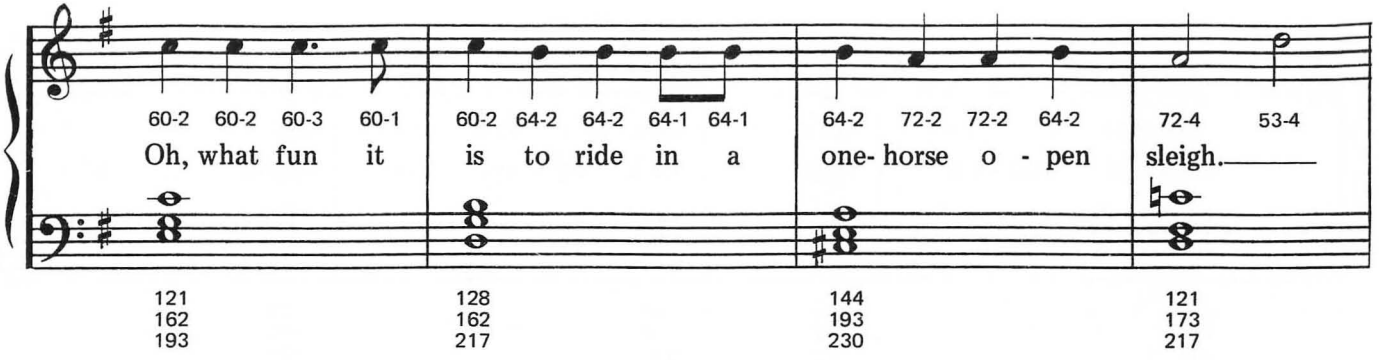
4. Some say he's a bad one,
   But I say he is bonnie
   Fairest of them all
   Is my handsome, winsome Johnny.

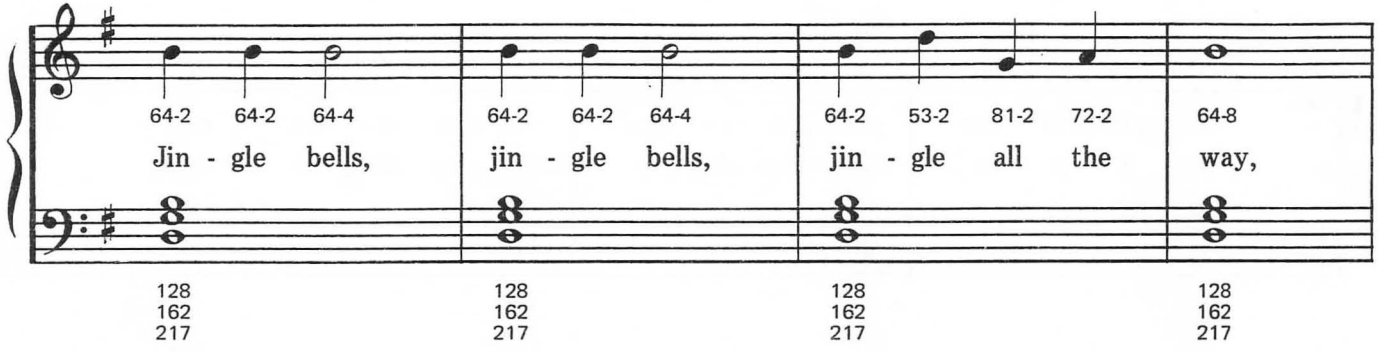
JINGLE BELLS

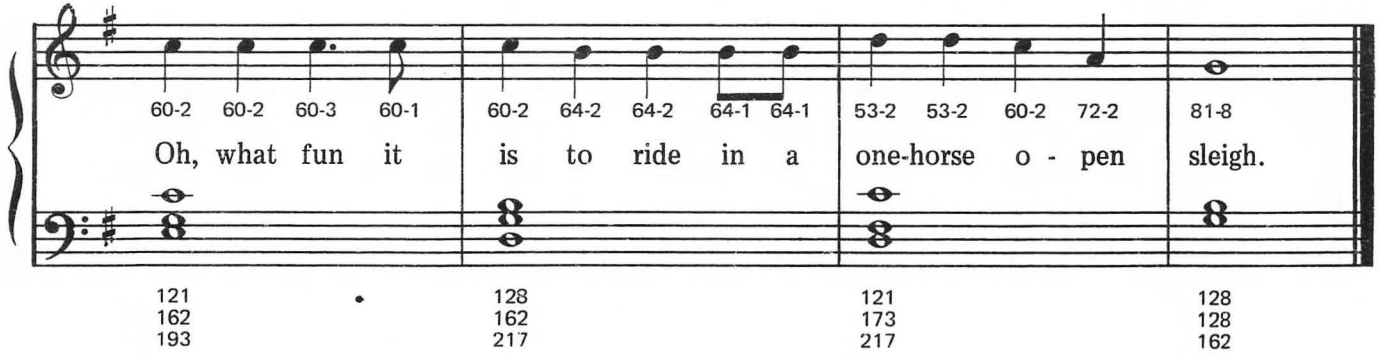
J. Pierpont




108-2 64-2 72-2 81-2
Bells on bob - tail
108-8
ring,
108-2 64-2 72-2 81-2
mak - ing spi - rits
96-6 96-2
bright, what
128 128 128 121
128 128 128 162
162 162 162 193
96-2 60-2 64-2 72-2
fun it is to
53-2 53-2 53-2 53-2
ride and sing a
47-2 53-2 60-2 72-2
sleigh - ing song to -
81-8
night.
121 121 121 128
144 173 173 128
193 217 217 162
64-2 64-2 64-4
Jin - gle bells,
64-2 64-2 64-4
jin - gle bells,
64-2 53-2 81-3 72-1
iin - gle all the
64-8
way,
128 128 128 128
162 162 162 162
217 217 217 217

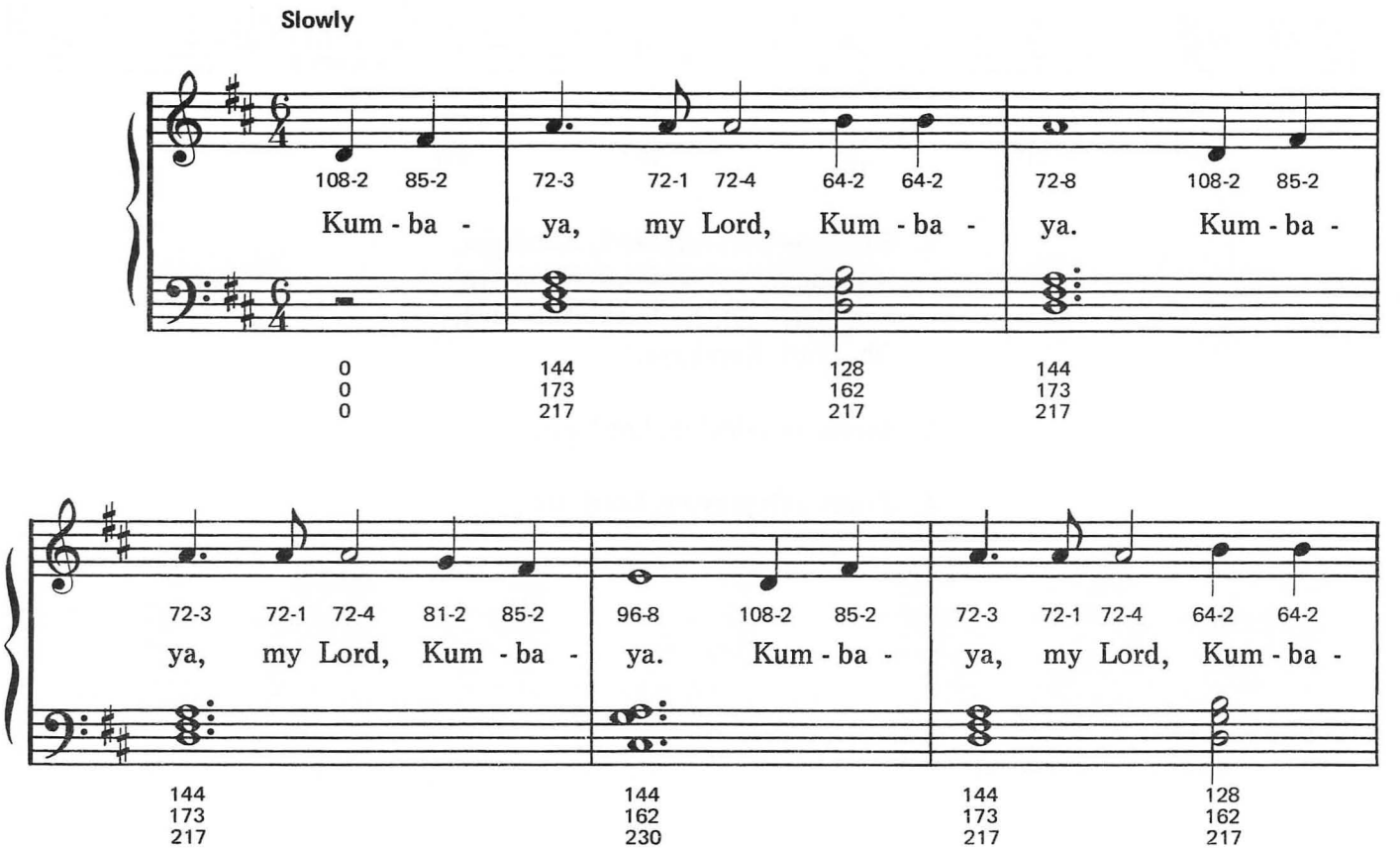

60-2 60-2 60-3 60-1 60-2 64-2 64-2 64-1 64-1 64-2 72-2 72-2 64-2 72-4 53-4
Oh, what fun it is to ride in a one- horse o - pen sleigh.
121 162 193
128 162 217
144 193 230
121 173 217

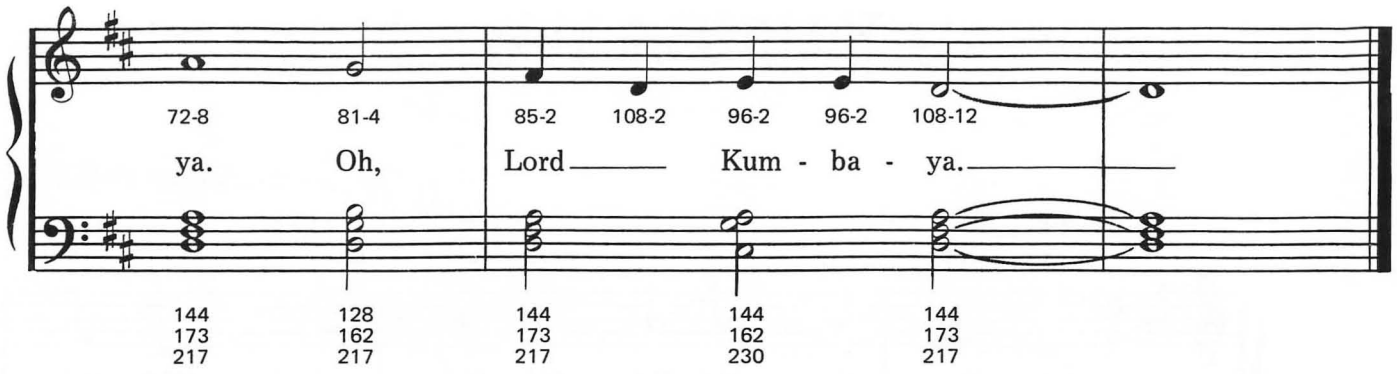

64-2 64-2 64-4 64-2 64-2 64-4 64-2 53-2 81-2 72-2 64-8
Jin - gle bells, jin - gle bells, jin - gle all the way,
128 162 217
128 162 217
128 162 217
128 162 217


60-2 60-2 60-3 60-1 60-2 64-2 64-2 64-1 64-1 53-2 53-2 60-2 72-2 81-8
Oh, what fun it is to ride in a one-horse o - pen sleigh.
121 162 193
128 162 217
121 173 217
128 128 162

KUMBAYA

African Folk Song





2. Someone's crying, Lord, Kumbaya,
Someone's crying, Lord, Kumbaya,
Someone's crying, Lord, Kumbaya,
Oh, Lord, Kumbaya.

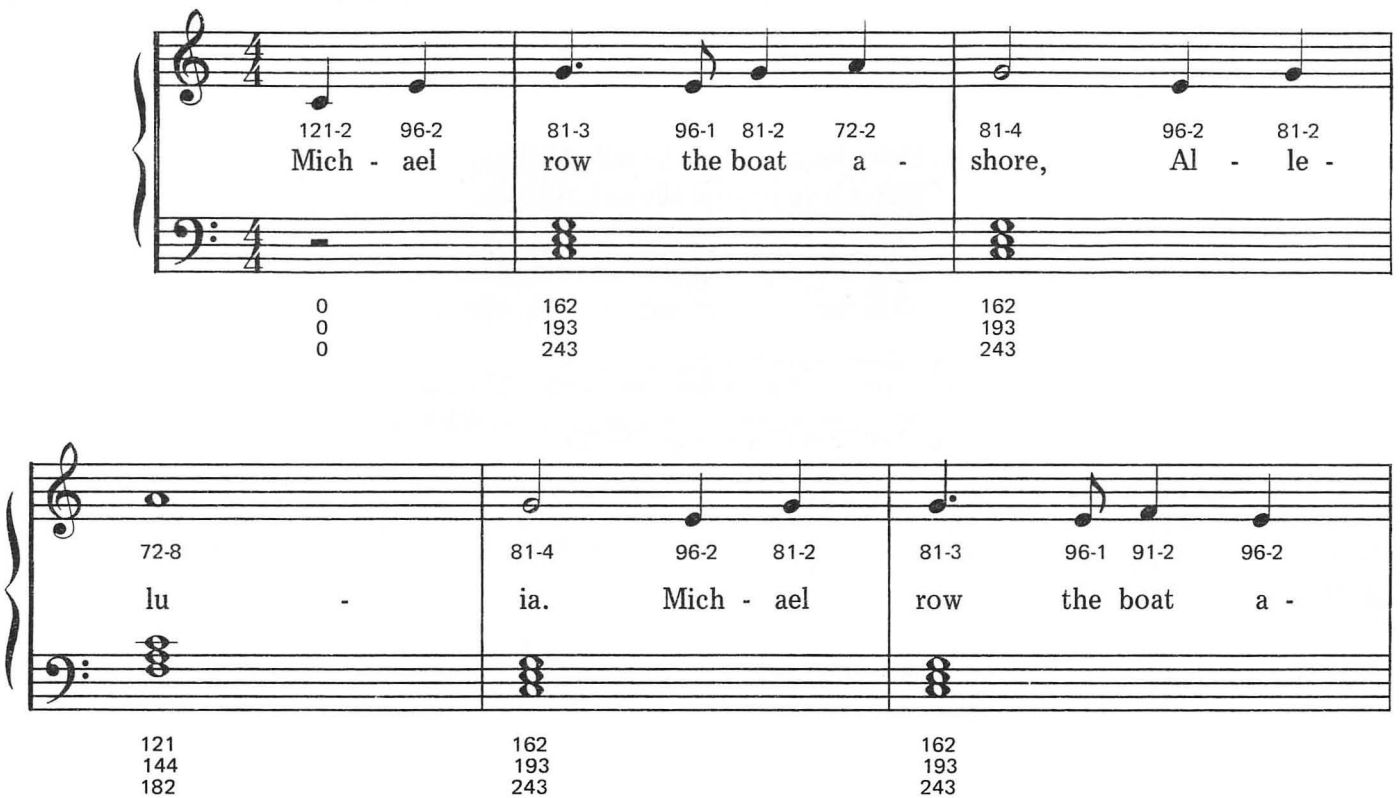
3. Someone's singing, Lord, *etc.*

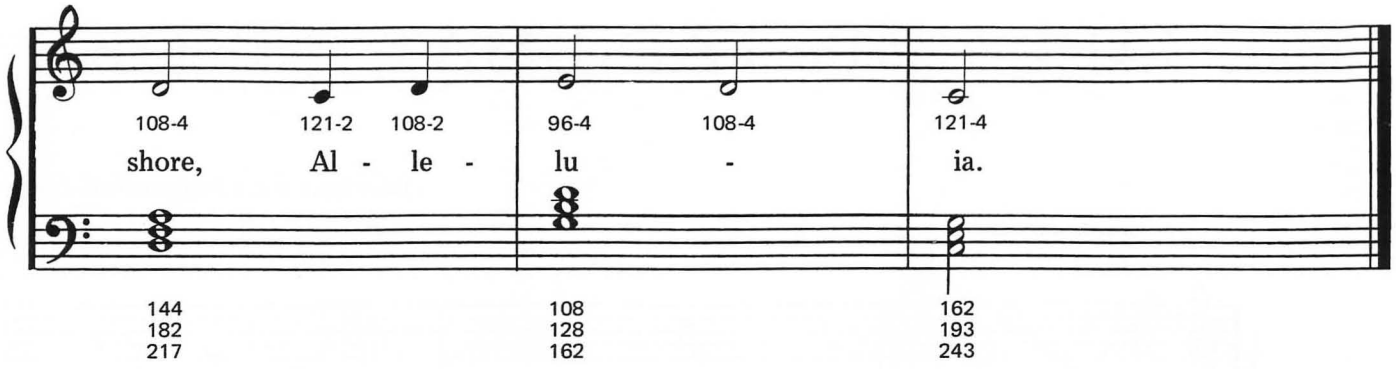
4. Someone's praying, Lord, *etc.*

MICHAEL ROW THE BOAT ASHORE

American West Indian Folk Song

Moderately Slow





2. Sister help to trim the sail, Alleluia,
Sister help to trim the sail, Alleluia.

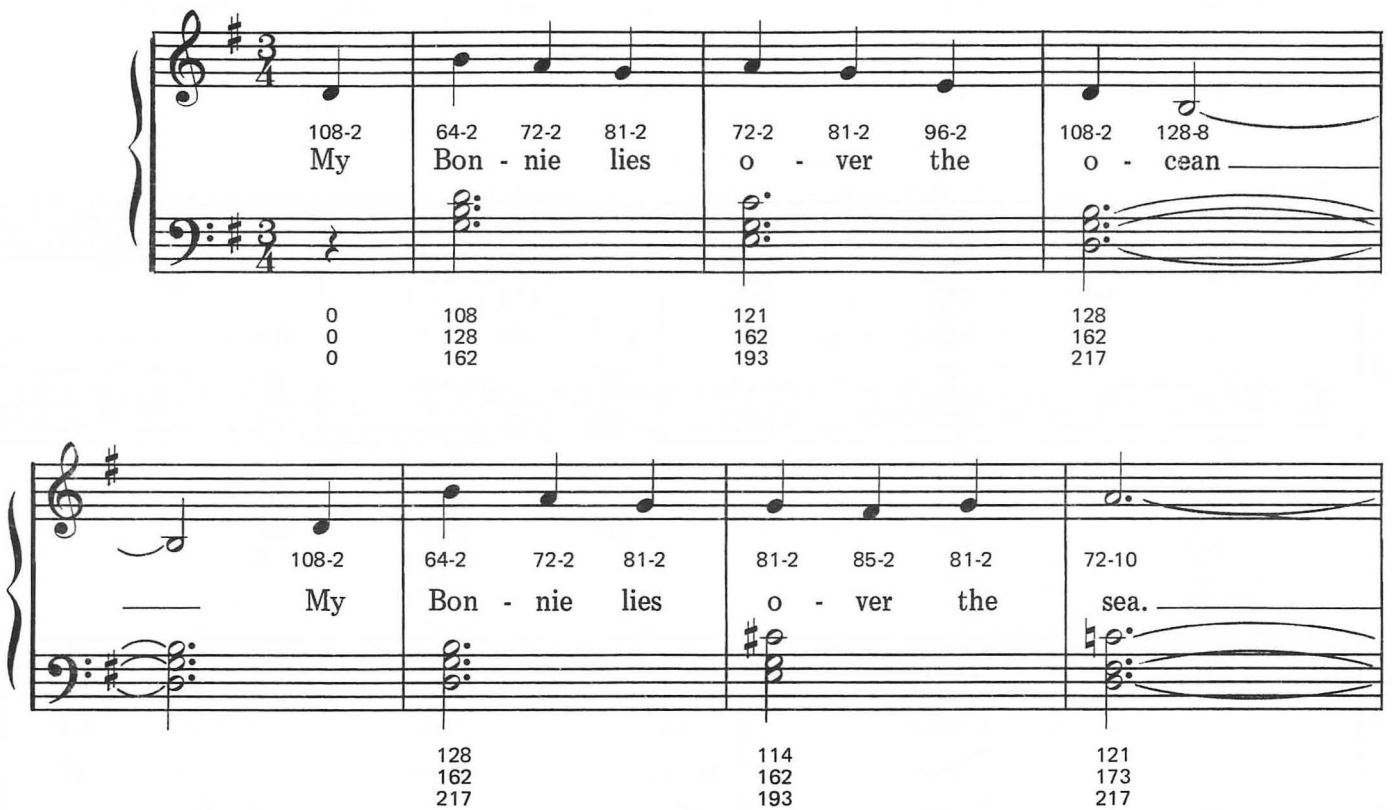
3. Jordan's river is chilly and cold, Alleluia,
Kills the body but not the soul, Alleluia.

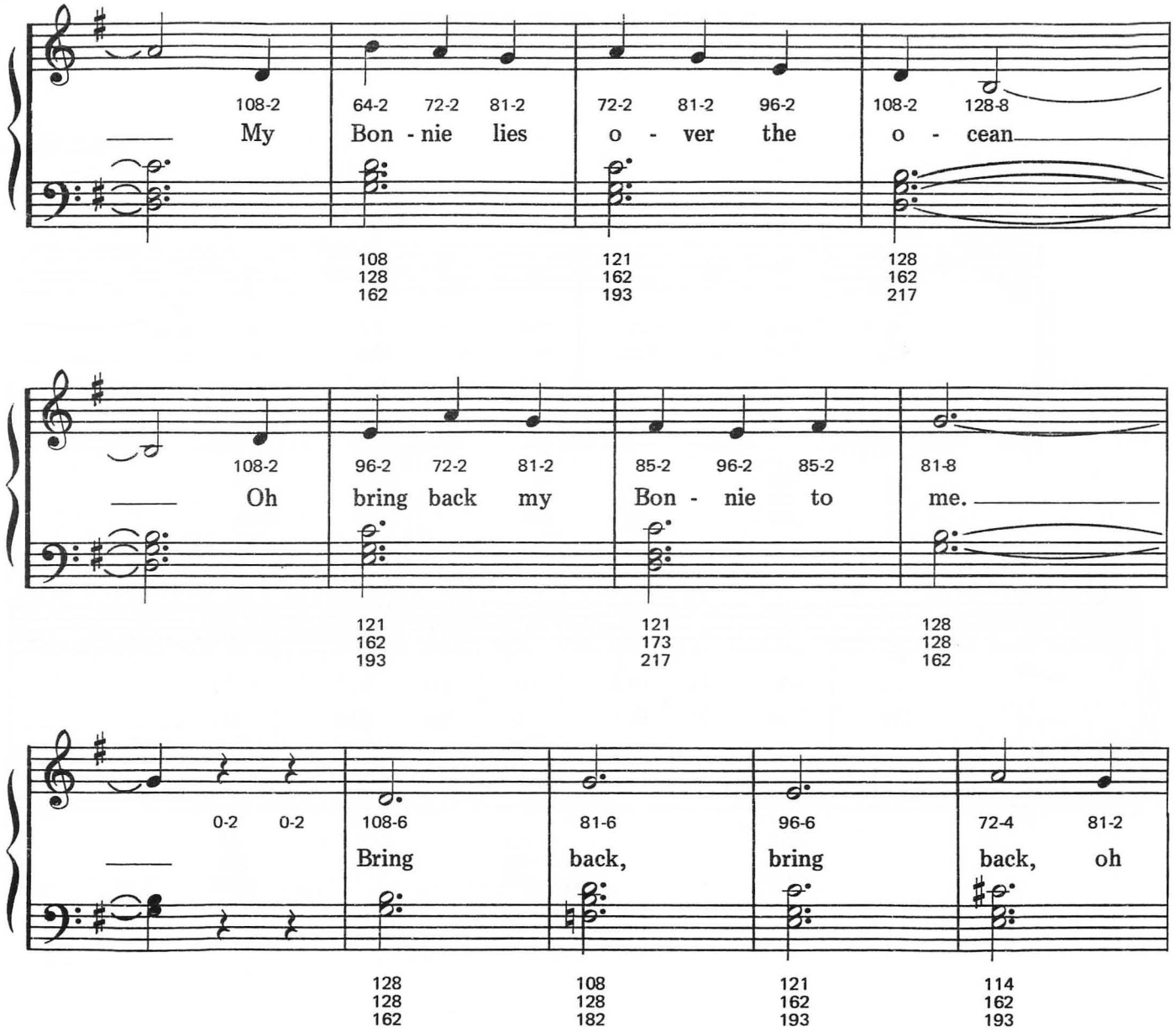
4. Jordan's river is deep and wide, Alleluia,
Milk and honey on the other side, Alleluia.

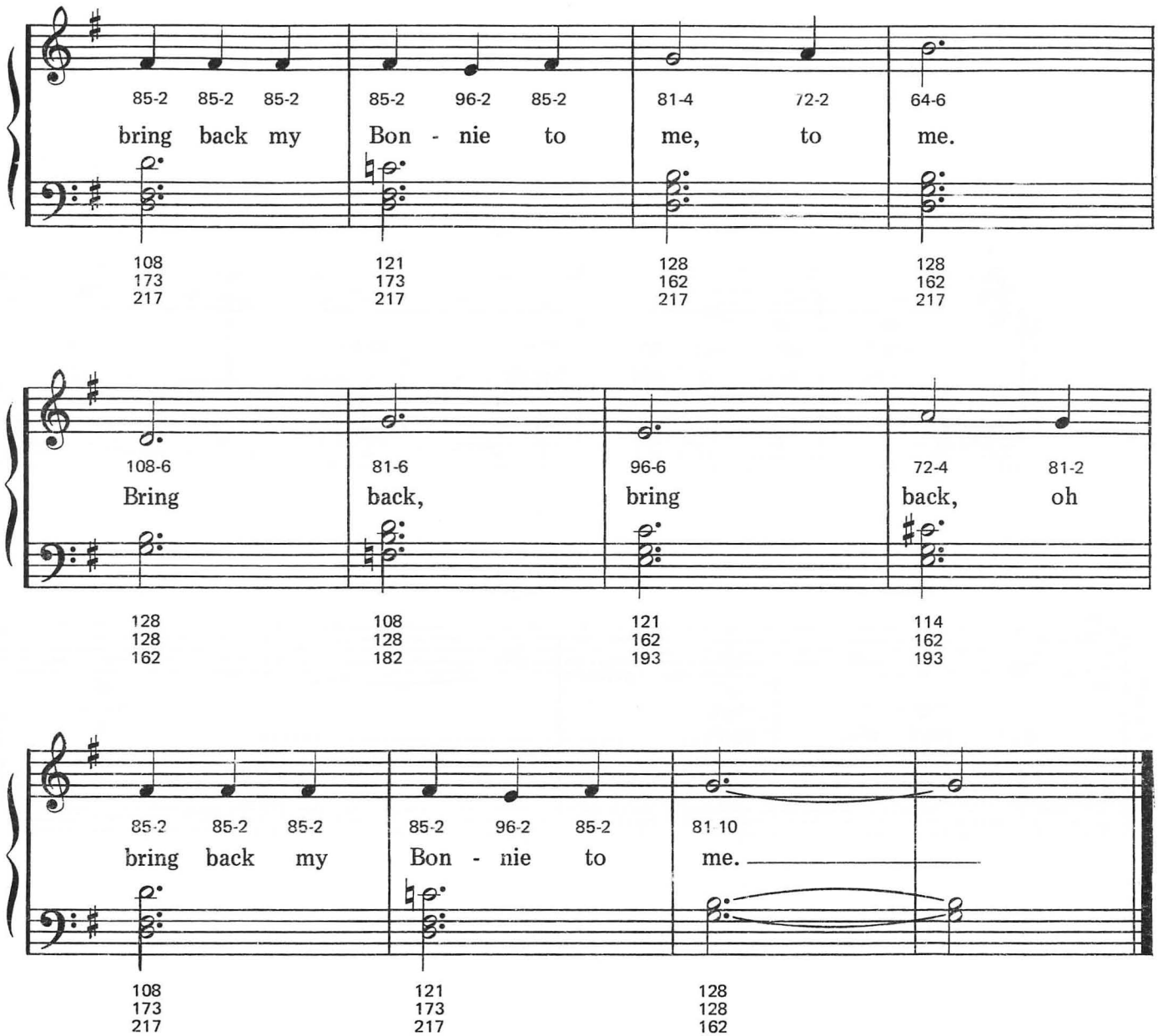
MY BONNIE

Traditional

Moderately

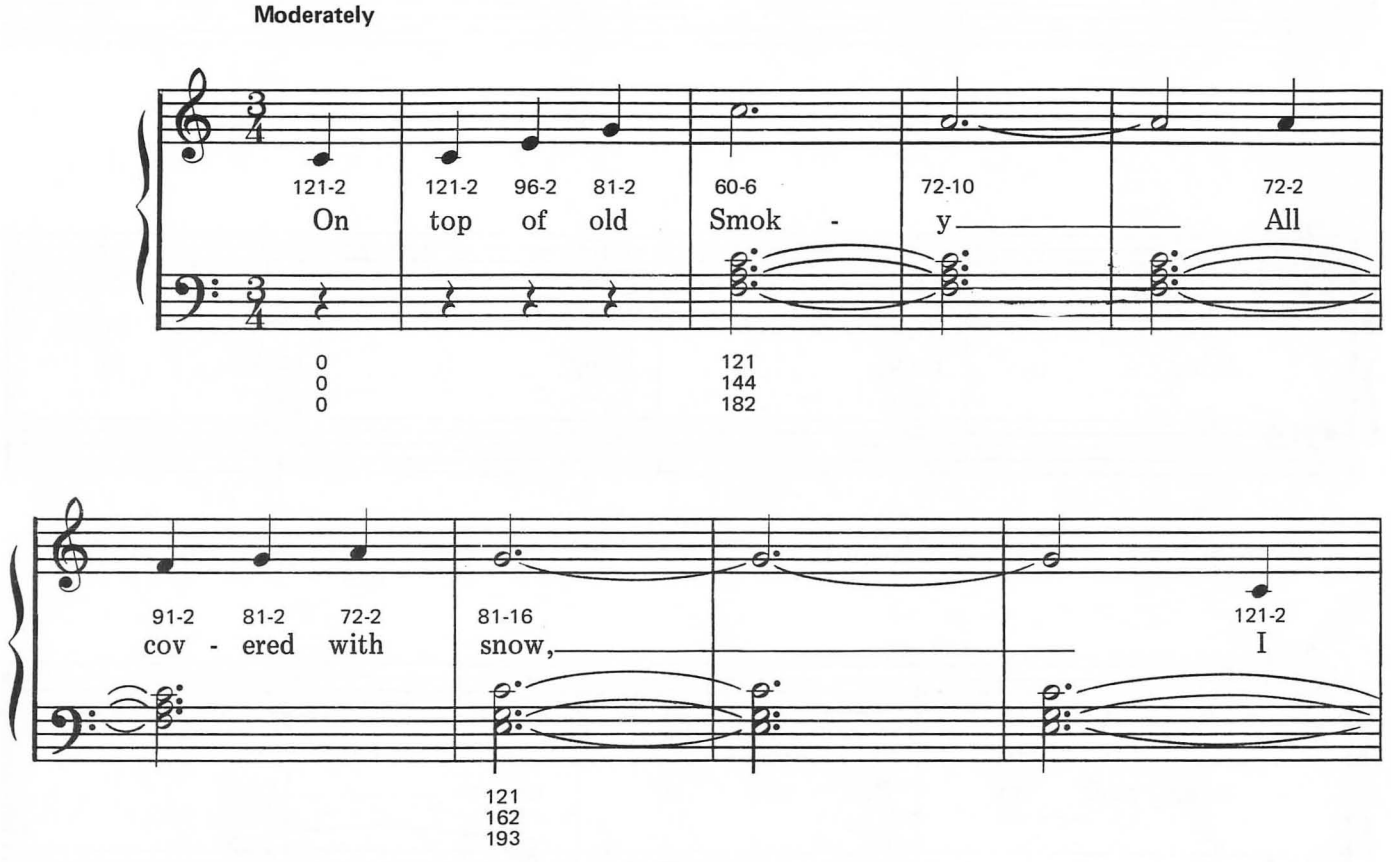


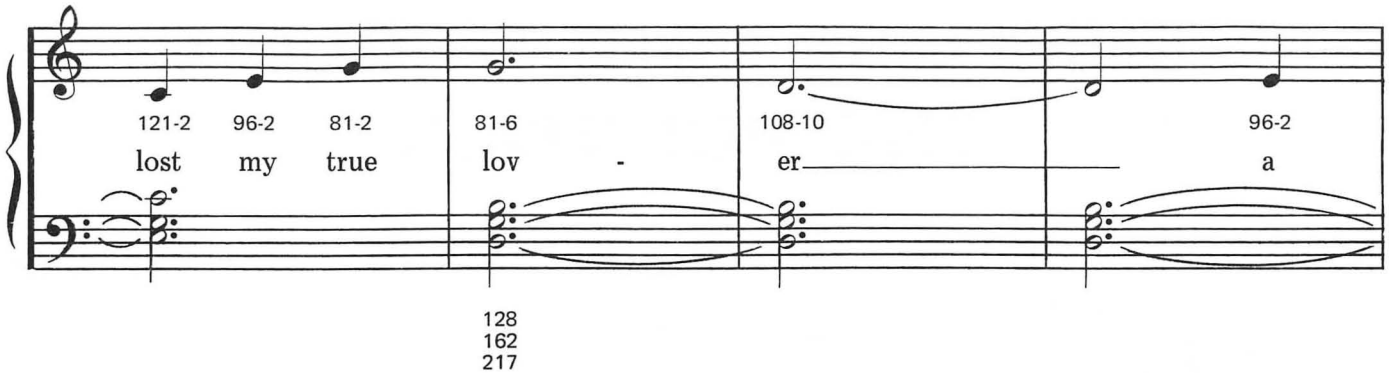

108-2 64-2 72-2 81-2 72-2 81-2 96-2 108-2 128-8
My Bon - nie lies o - ver the o - cean
108 128 162 121 162 193 128 162 217
108-2 96-2 72-2 81-2 85-2 96-2 85-2 81-8
Oh bring back my Bon - nie to me.
121 162 193 121 173 217 128 128 162
0-2 0-2 108-6 81-6 96-6 72-4 81-2
Bring back, bring back, oh
128 128 162 108 128 182 121 162 193 114 162 193

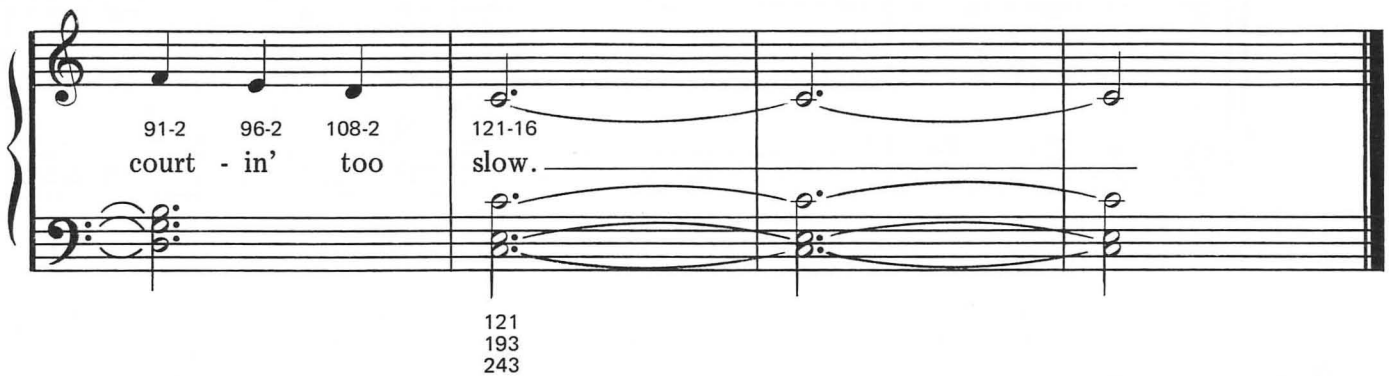

85-2 85-2 85-2 85-2 96-2 85-2 81-4 72-2 64-6
bring back my Bon - nie to me, to me.
108 173 217
121 173 217
128 162 217
128 162 217
108-6 81-6 96-6 72-4 81-2
Bring back, bring back, oh
128 128 162
108 128 182
121 162 193
114 162 193
85-2 85-2 85-2 85-2 96-2 85-2 81 10
bring back my Bon - nie to me.
108 173 217
121 173 217
128 128 162

ON TOP OF OLD SMOKEY

American Folk Song







2. A-courtin's a pleasure
A-flirtin's a grief,
A false-hearted lover,
Is worse than a thief.

3. A thief will just rob you,
And take what you have.
But a false-hearted lover,
Will lead you to the grave.

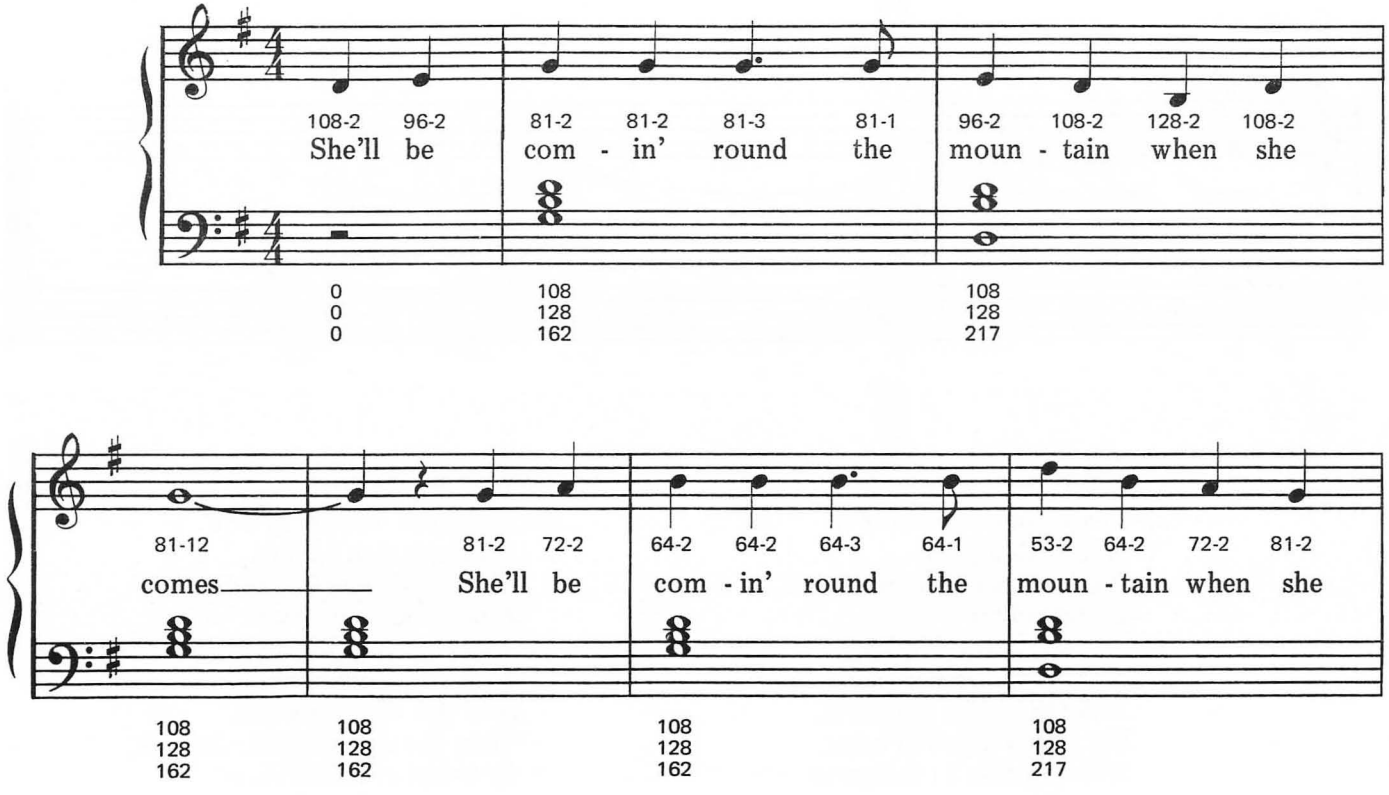
4. The grave will decay you,
And turn you to dust,
There ain't one in a million,
A poor boy can trust.

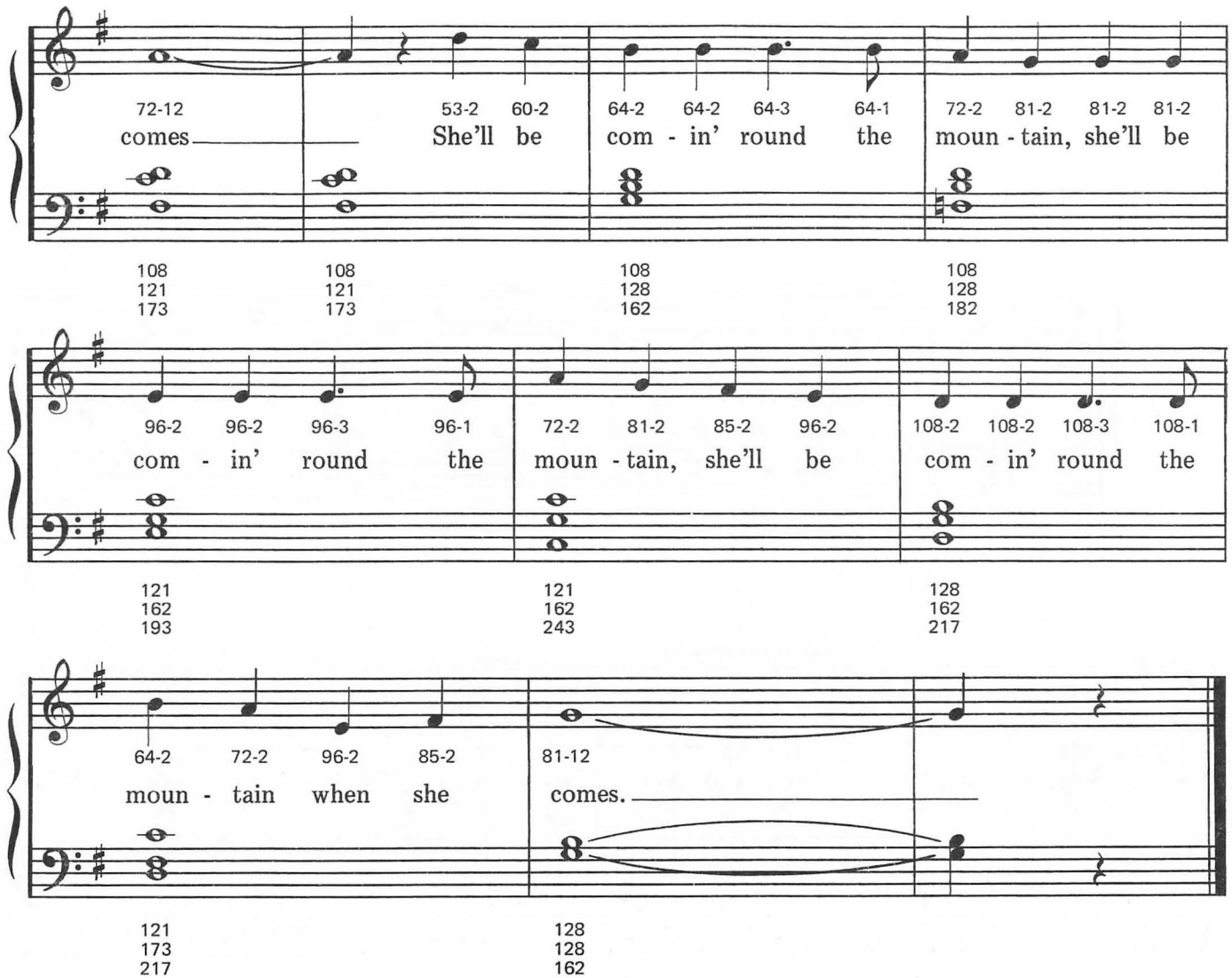
5. They'll hug you and kiss you,
And tell you more lies,
Than the crossties on railroads,
Or the stars in the skies.

SHE'LL BE COMIN' ROUND THE MOUNTAIN

Traditional

Moderately

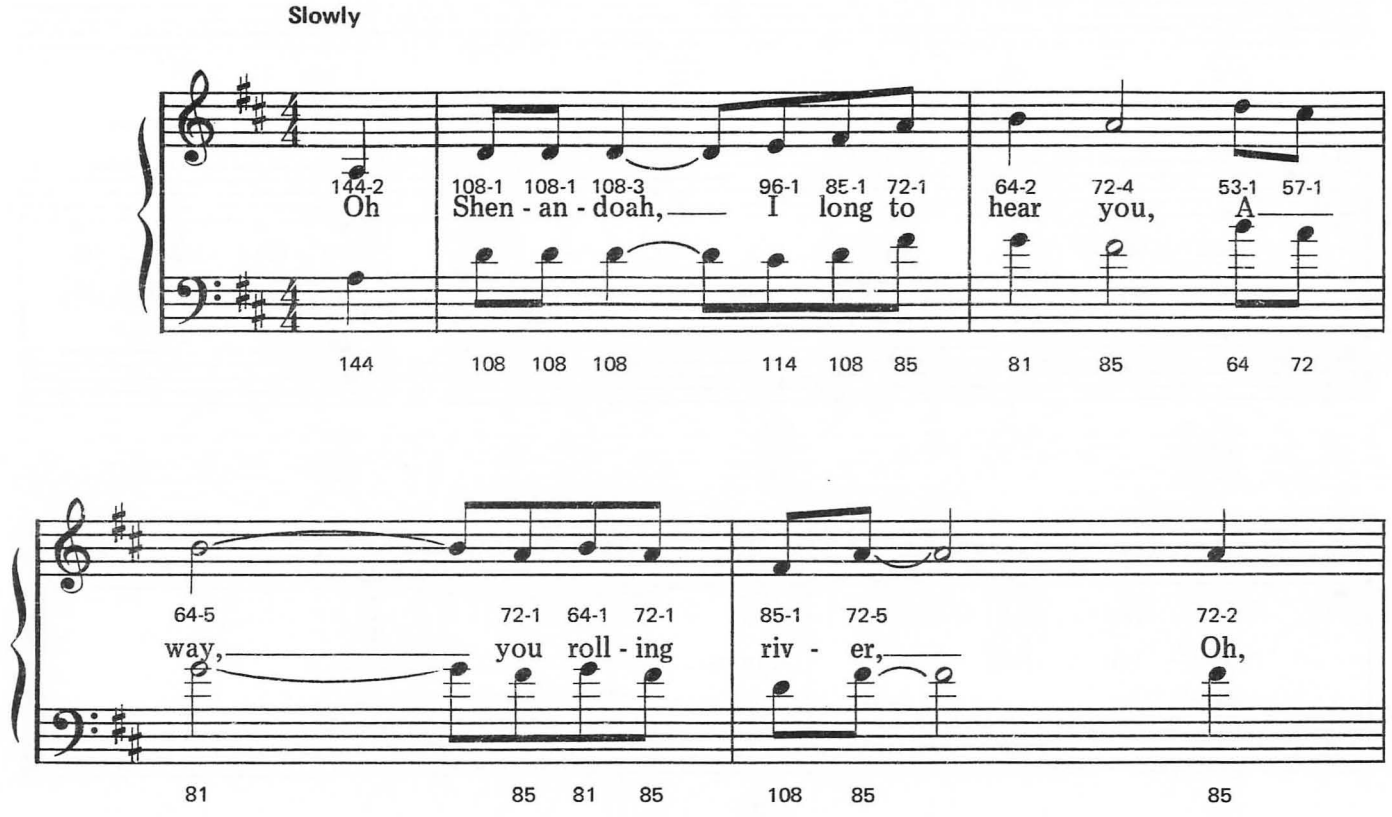


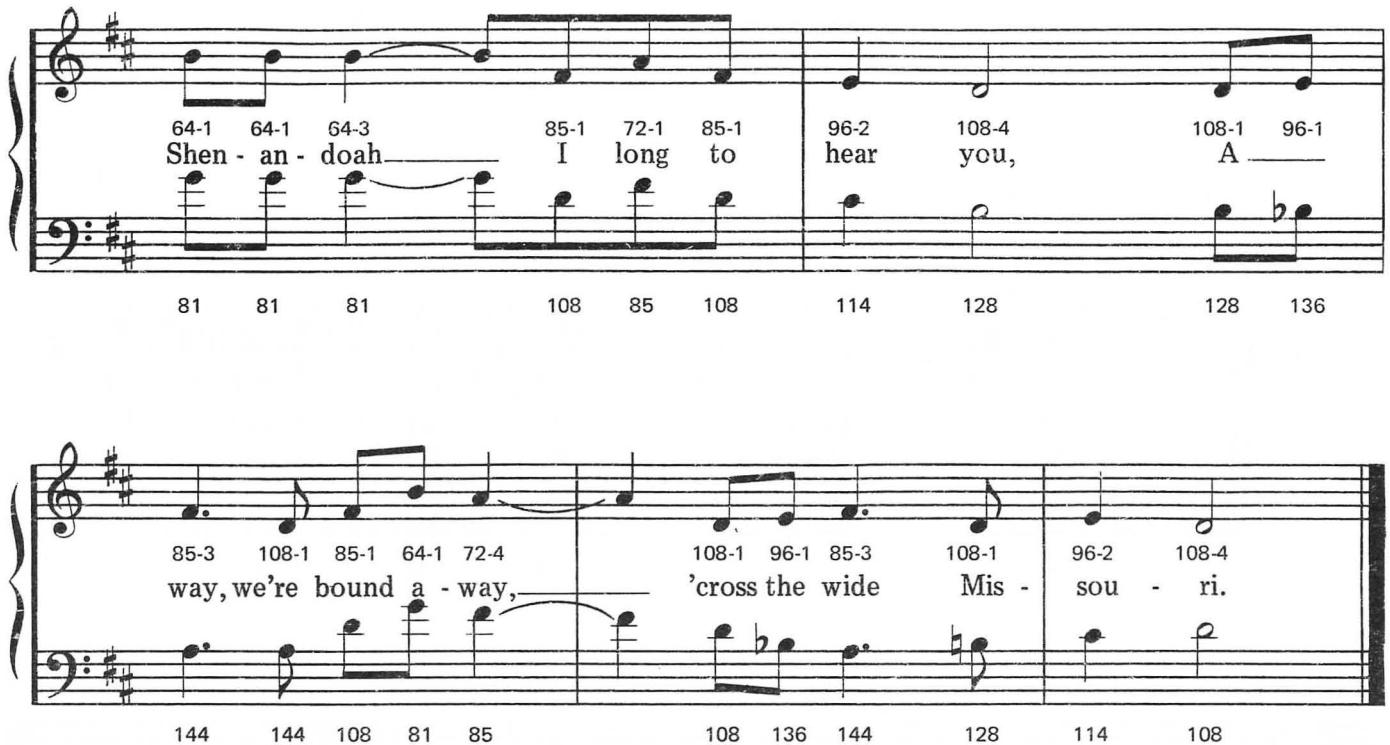


2. She'll be drivin' six white horses when she comes, *etc.*
3. Oh, we'll all go out to meet her when she comes, *etc.*
4. Oh, we'll all have chicken and dumplings when she comes, *etc.*

SHENANDOAH

American Sea Chanty





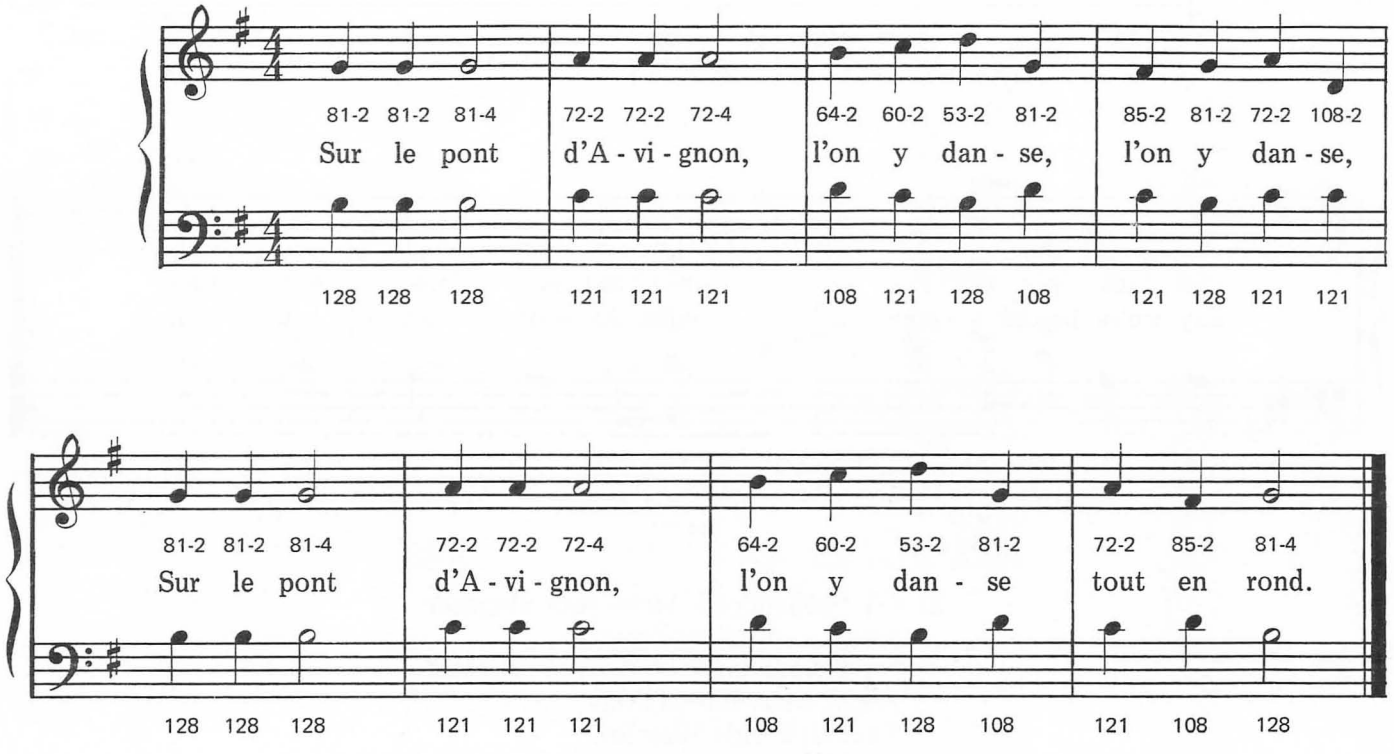
2. Oh, Shenandoah, I love your daughter,
Away, you rolling river.
Oh, Shenandoah, I love your daughter,
Away, we're bound away,
'Cross the wide Missouri.

3. Oh, Shenandoah, I'm bound to leave you,
Away, you rolling river.
Oh, Shenandoah, I'm bound to leave you,
Away, we're bound away,
'Cross the wide Missouri.

SUR LE PONT D'AVIGNON

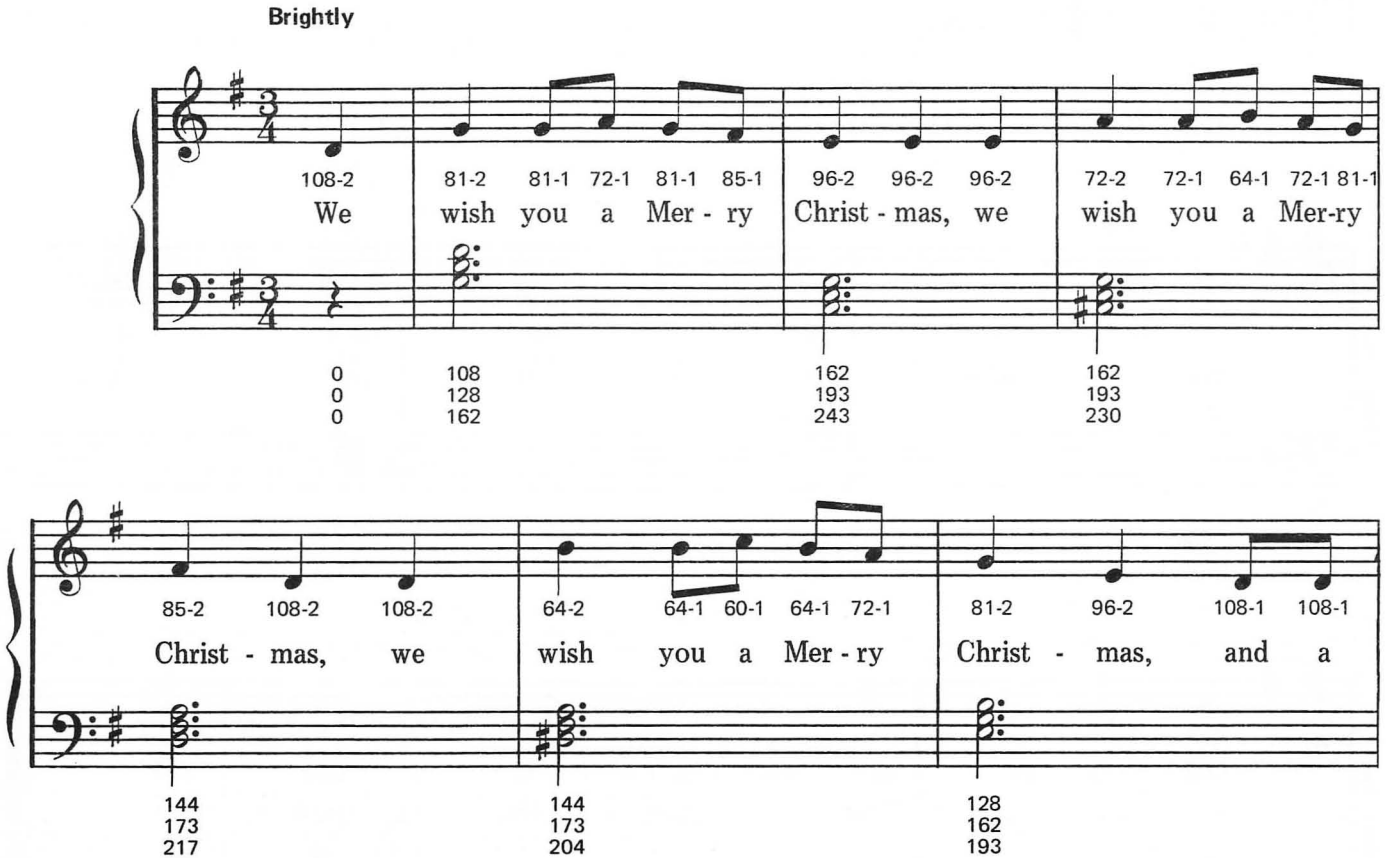
French Folk Song

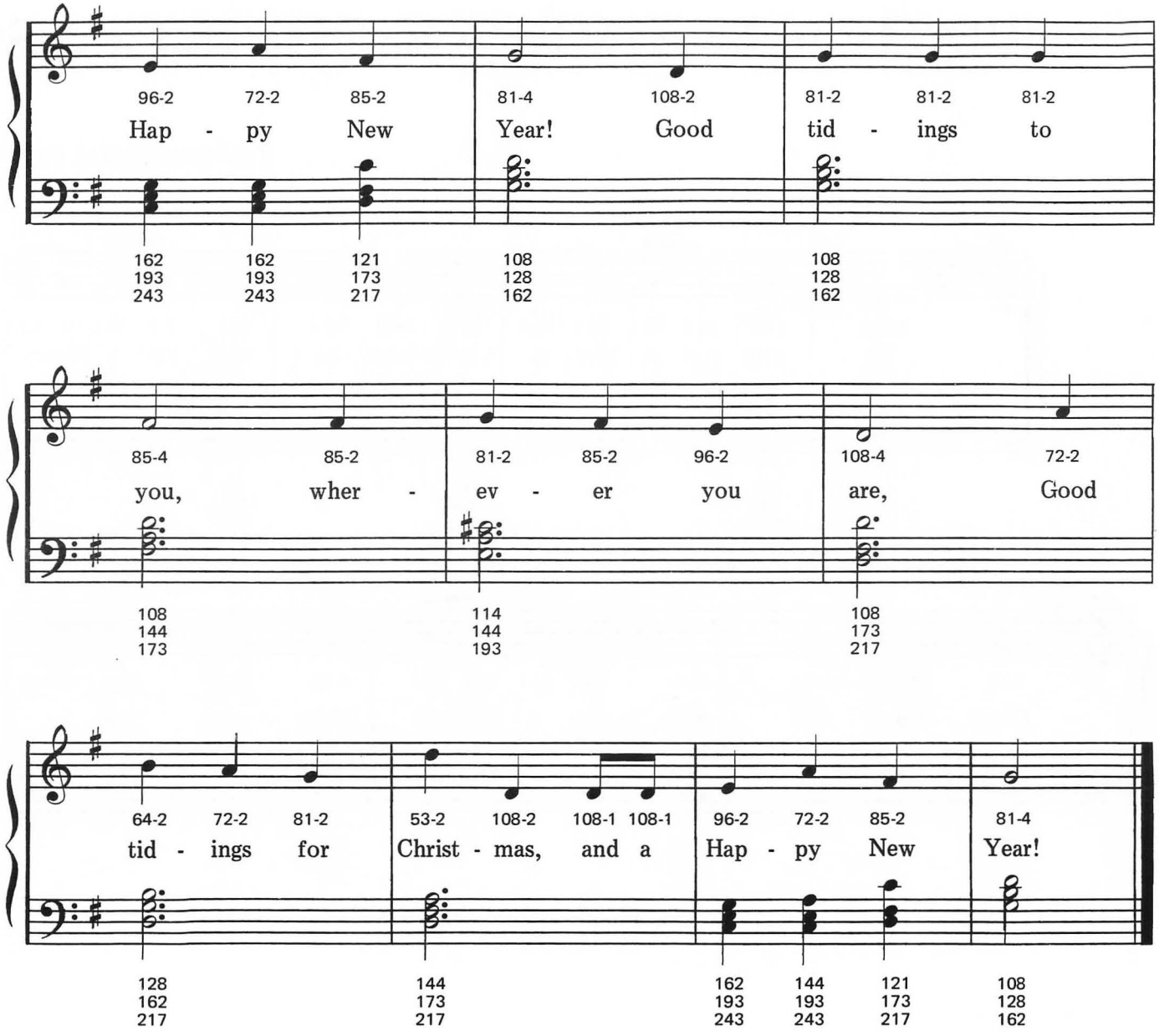
Moderately



WE WISH YOU A MERRY CHRISTMAS

Traditional English Carol

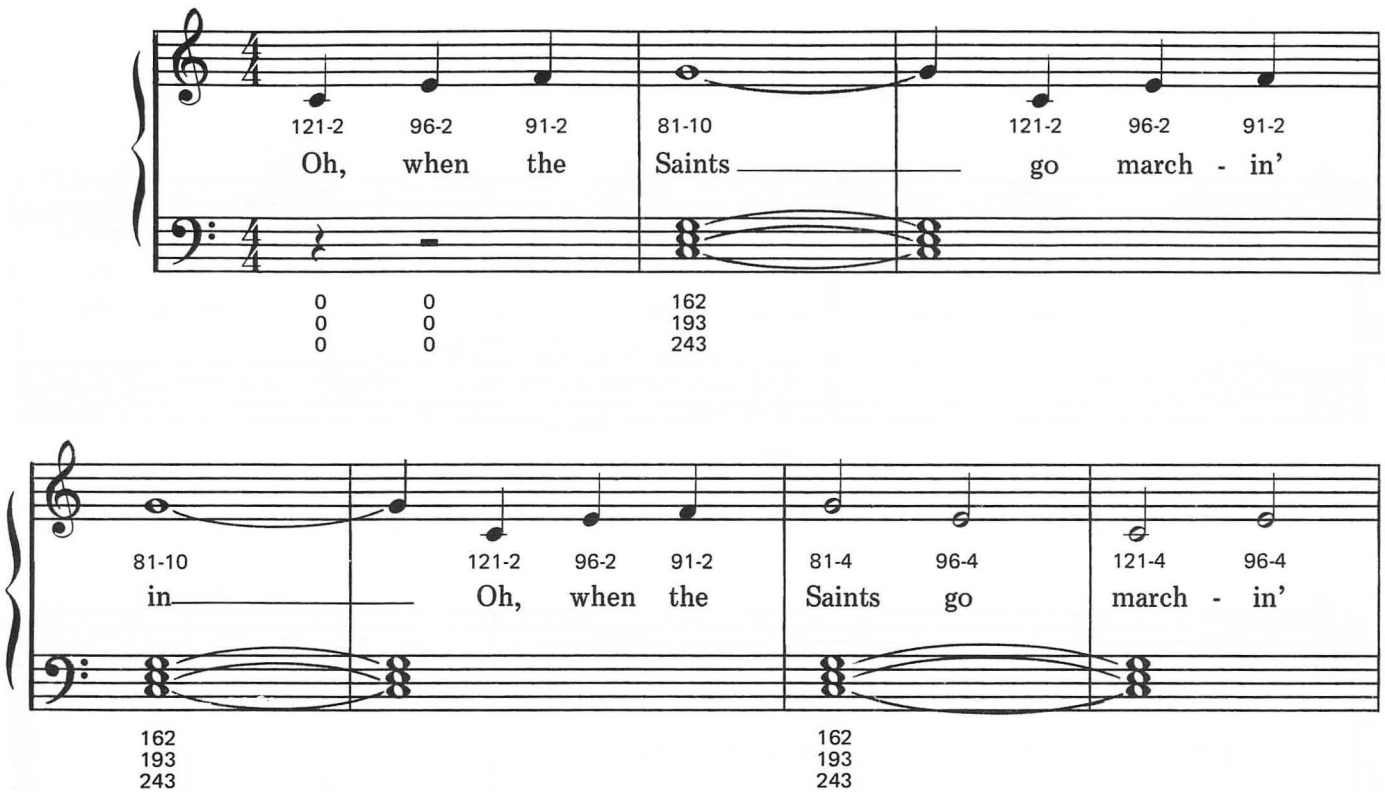


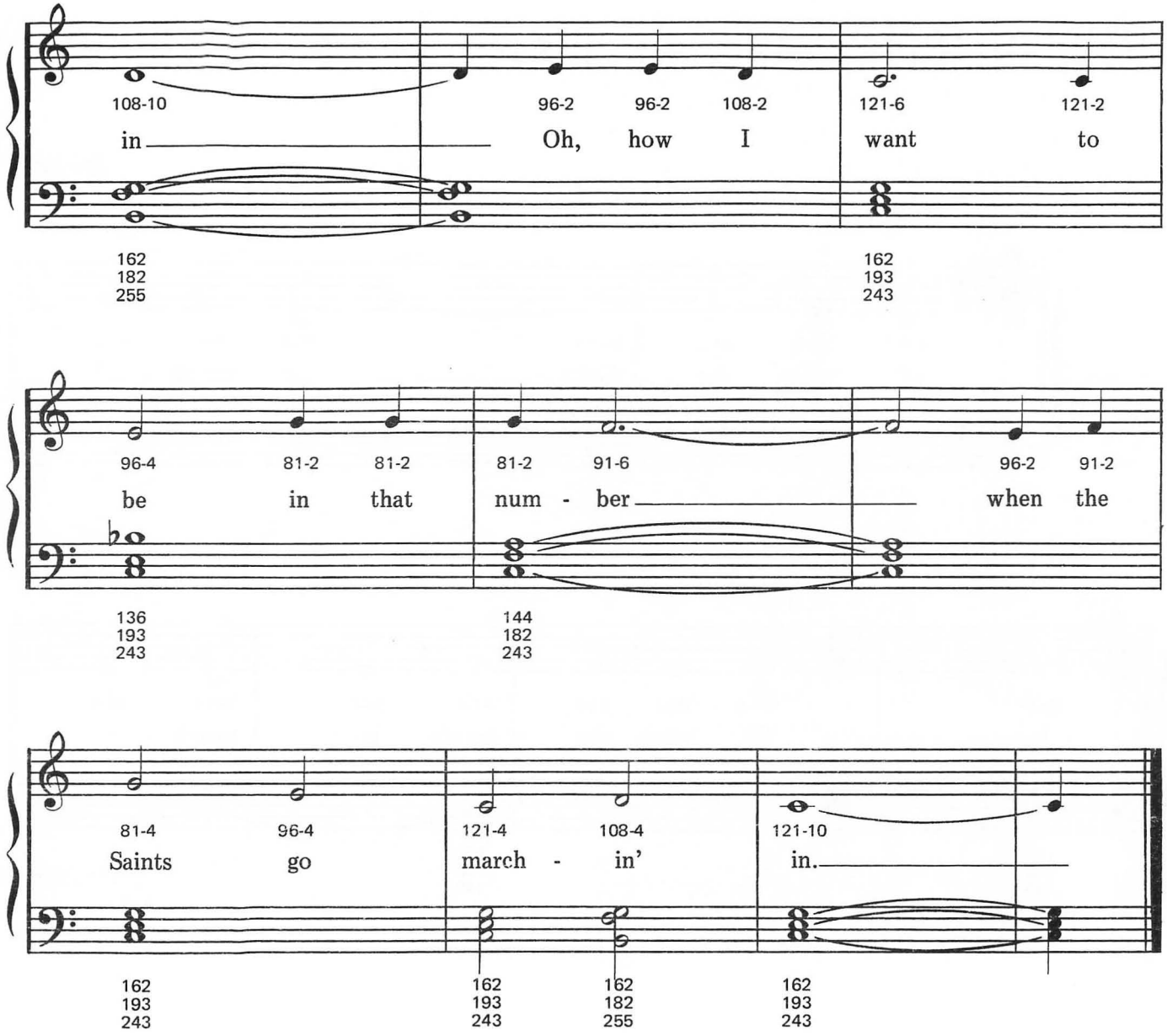

96-2 72-2 85-2 81-4 108-2 81-2 81-2 81-2
Hap - py New Year! Good tid - ings to
162 162 121 108 108
193 193 173 128 128
243 243 217 162 162
85-4 85-2 81-2 85-2 96-2 108-4 72-2
you, wher - ev - er you are, Good
108 114 108
144 144 173
173 193 217
64-2 72-2 81-2 53-2 108-2 108-1 108-1 96-2 72-2 85-2 81-4
tid - ings for Christ - mas, and a Hap - py New Year!
128 144 162 144 121 108
162 173 193 193 173 128
217 217 243 243 217 162

WHEN THE SAINTS GO MARCHING IN

Spiritual

Moderately Fast

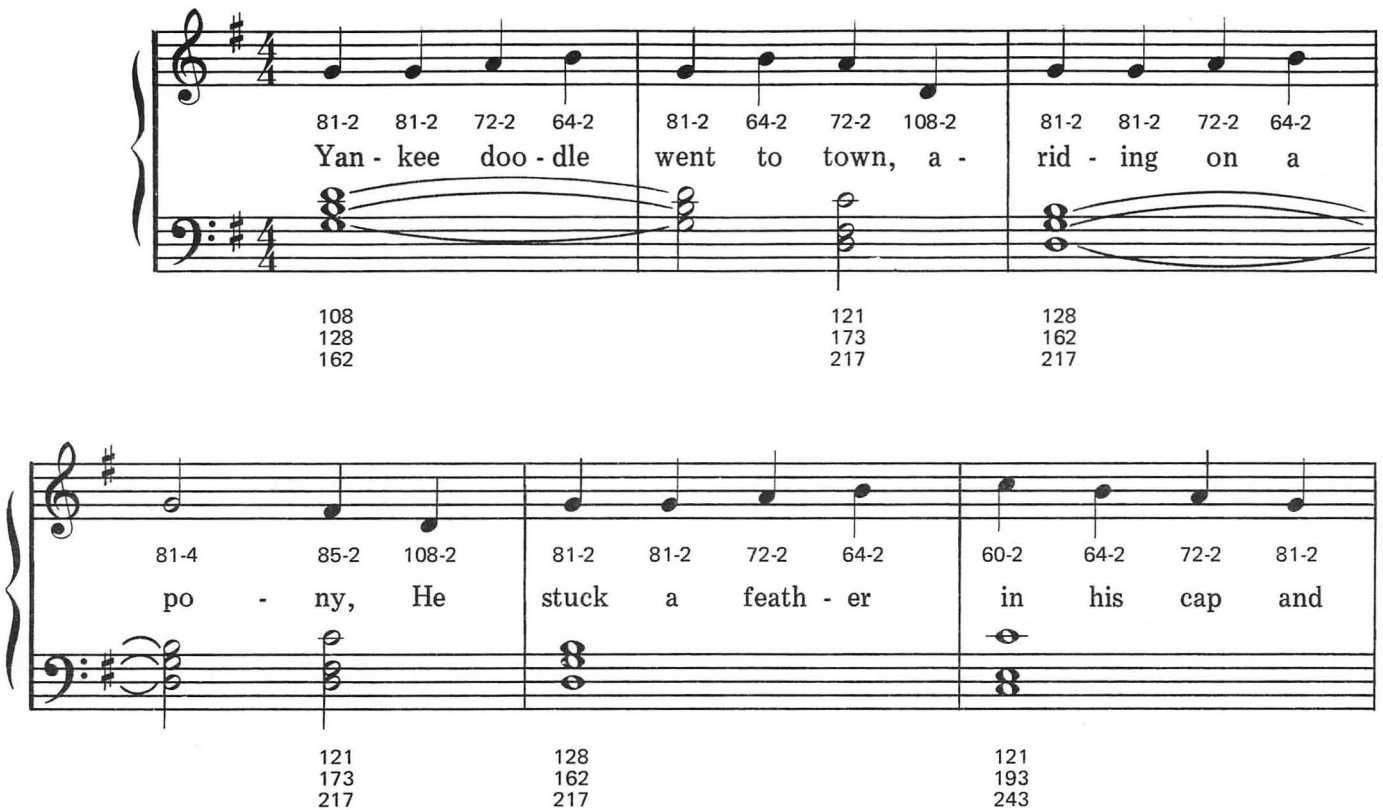


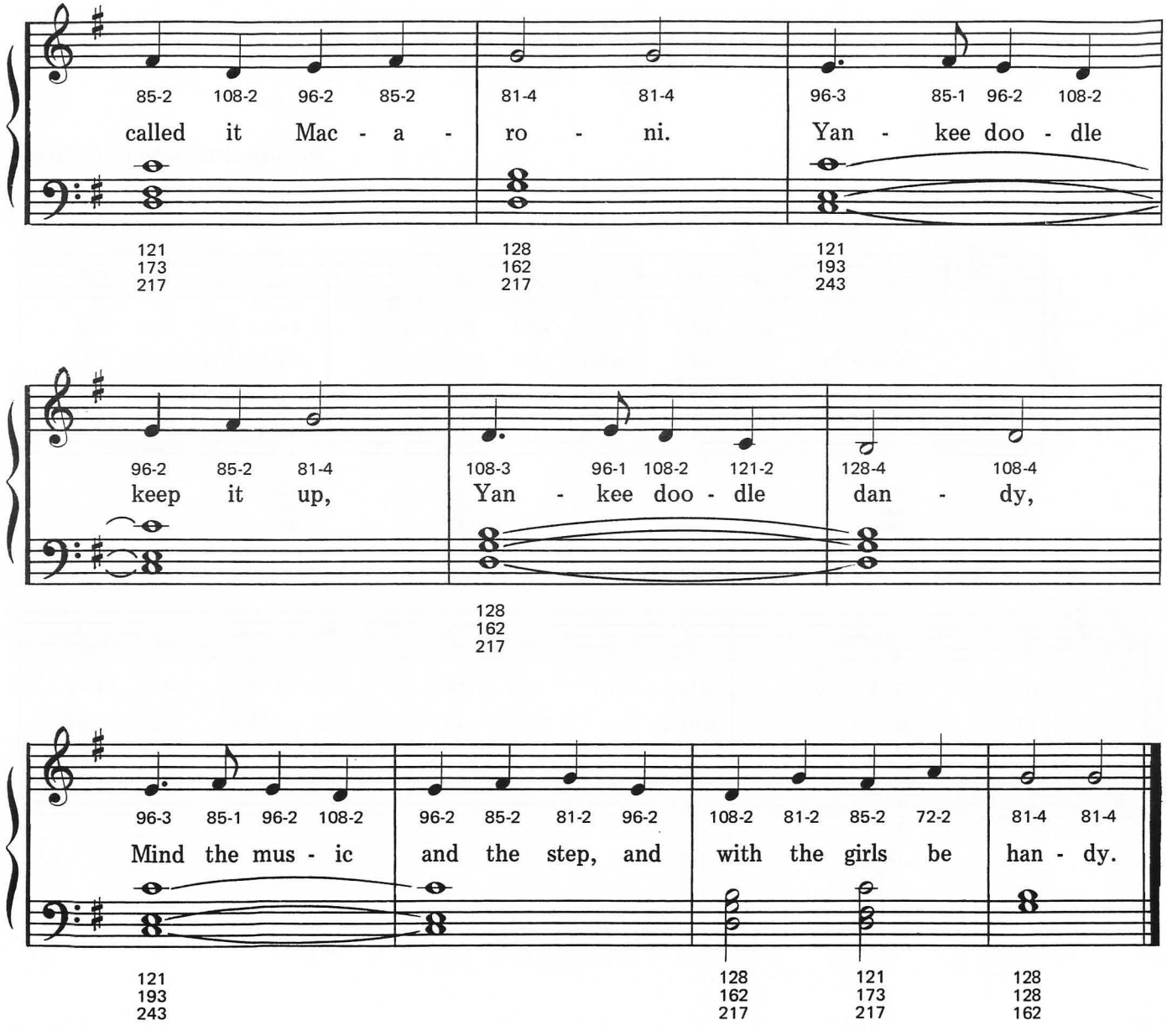

108-10
in
96-2
96-2
108-2
Oh, how I
121-6
121-2
want to
162
182
255
162
193
243
96-4
81-2
81-2
be in that
81-2
91-6
num - ber
96-2
91-2
when the
136
193
243
144
182
243
81-4
96-4
Saints go
121-4
108-4
march - in'
121-10
in.
162
193
243
162
193
243
162
182
255
162
193
243

YANKEE DOODLE

English-American Folk Song

Moderately

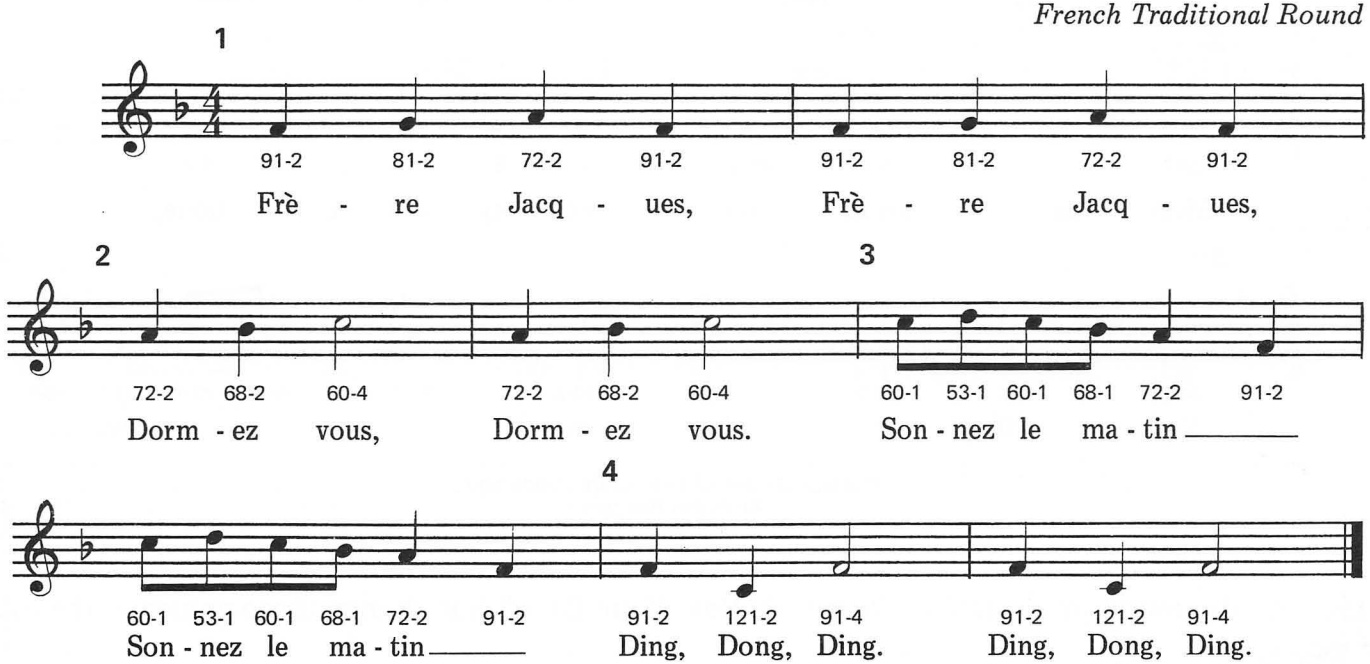



85-2 108-2 96-2 85-2 81-4 81-4 96-3 85-1 96-2 108-2
called it Mac - a - ro - ni. Yan - kee doo - dle
121 173 217
128 162 217
121 193 243
96-2 85-2 81-4 108-3 96-1 108-2 121-2 128-4 108-4
keep it up, Yan - kee doo - dle dan - dy,
128 162 217
96-3 85-1 96-2 108-2 96-2 85-2 81-2 96-2 108-2 81-2 85-2 72-2 81-4 81-4
Mind the mus - ic and the step, and with the girls be han - dy.
121 193 243
128 162 217
121 173 217
128 128 162

13
ROUNDS

FRÈRE JACQUES

French Traditional Round

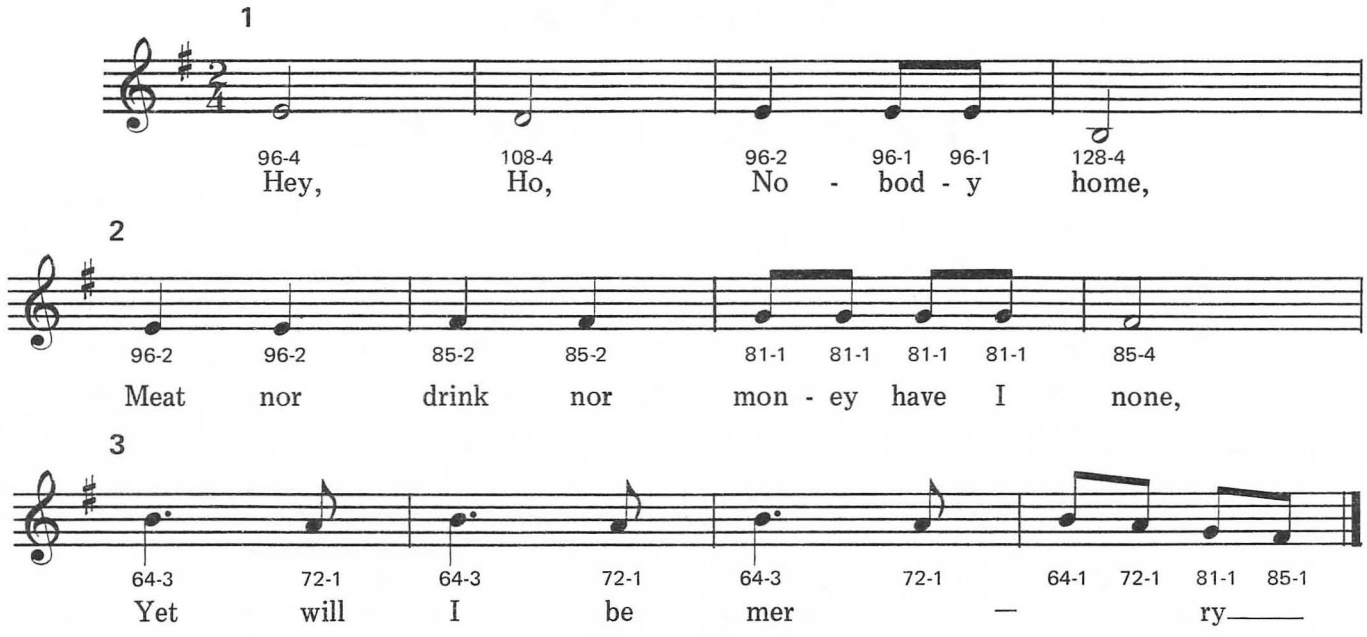


Start with the program for "Row Row Row Your Boat," but change these values in the round program:

```
370 F=16:REM OFFSET OF ROUND REPEAT
710 IF B>196 THEN END
```

HEY, HO, NOBODY HOME

English Traditional Round

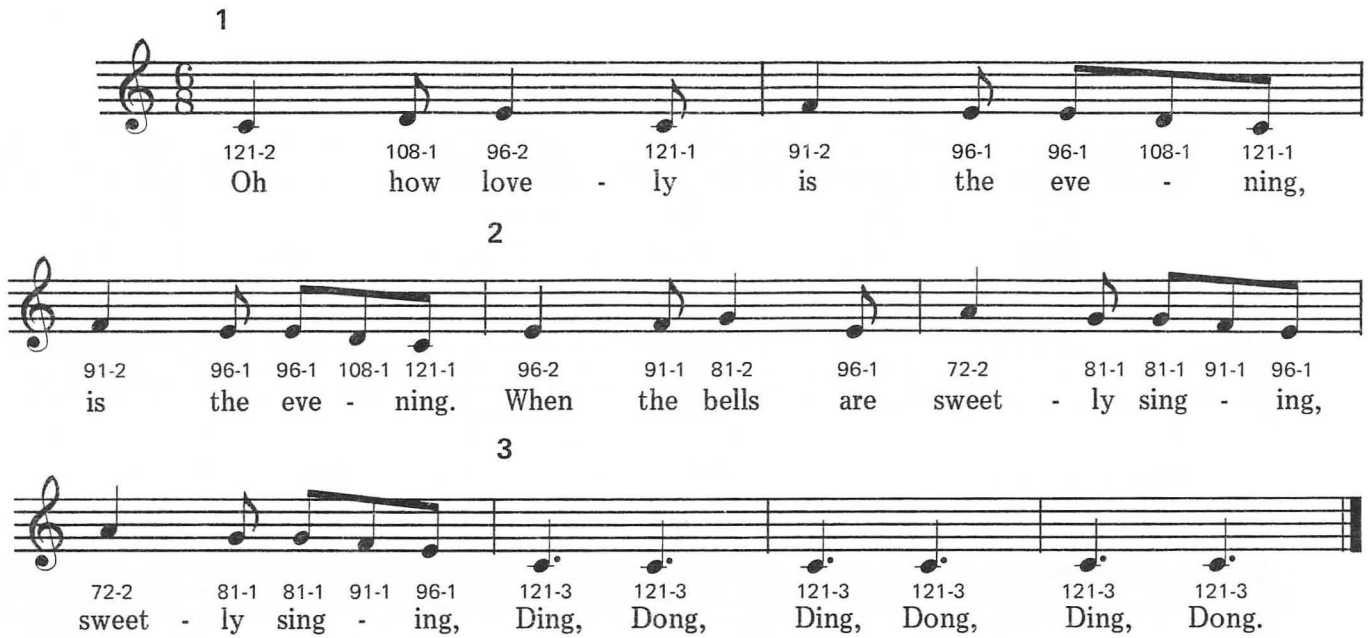


Start with the round program for "Row Row Row Your Boat," but change these values in the round program:

```
370 F=32:REM OFFSET OF ROUND REPEAT
710 IF B>196 THEN END
```

OH HOW LOVELY IS THE EVENING

English Traditional Round



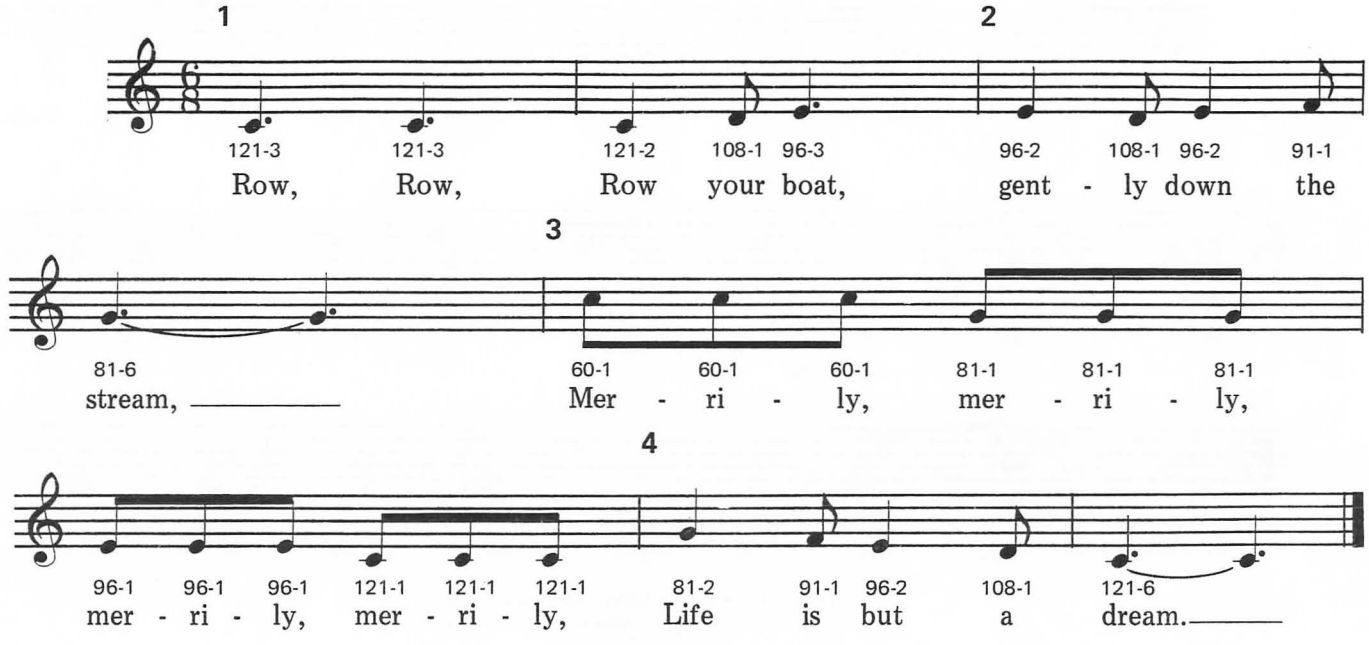
Start with the round program for "Row Row Row Your Boat," but change these values in the round program:

```
230 DIM A(224)
370 F=36:REM OFFSET OF ROUND REPEAT
450 IFZ= 220 THEN GOTO 490
710 IF B>222 THEN END
```

Add plenty of zeros to the DATA statements until there are fourteen rows of twelve each.

ROW, ROW, ROW YOUR BOAT

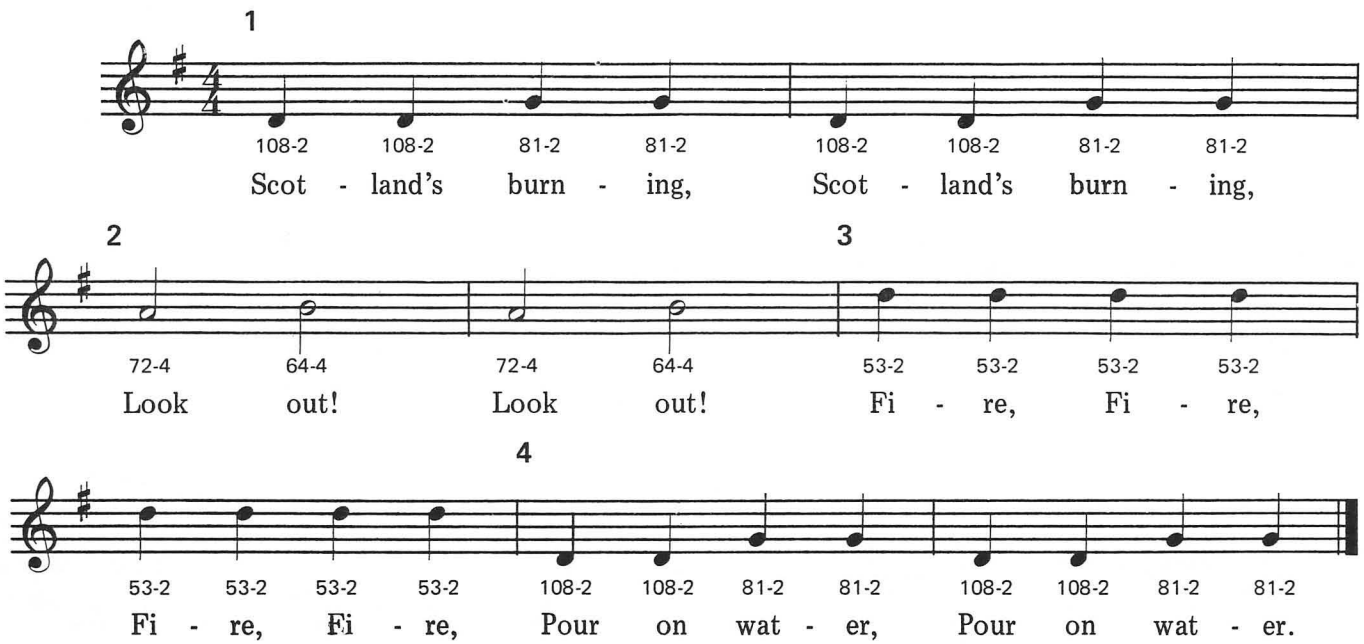
English Traditional Round



The entire program to play this song is written out in the chapter on rounds.

SCOTLAND'S BURNING

English Traditional Round



Start with the round program for "Row Row Row Your Boat," but change these values in the round program:

```
230 DIM A(224)
370 F=32:REM OFFSET OF ROUND REPEAT
450 IFZ= 220 THEN GOTO 490
710 IF B>222 THEN END
```

And plenty of zeros to the DATA statements until there are twelve rows of twelve each.

GLOSSARY OF MUSICAL TERMS

ACCENT The stress or emphasis on a note.

AMPLIFY To make a sound louder.

AMPLITUDE The measure of the loudness of a sound.

BAR LINES Vertical lines dividing the musical staff into measures.

BASS CLEF The staff of notes below middle C; usually used for the harmony.

BEAT A rhythmic pulse or emphasis of one or more notes in each measure.

CHORD A combination of notes that blend harmoniously when played together.

CLEF SIGN A symbol placed at the beginning of a musical staff which identifies the names of the lines and spaces of the staff.

DISTORTION Inaccurate reproduction of sound caused by changes which occur in transmission of the sound wave or electrical signal. In the Atari the distortion command causes the introduction of random noise into the pure tone.

DYNAMICS Musical contrast, changes in the force or intensity of the music.

FREQUENCY The number of times per second the air moves back and forth as a sound moves through it. Frequency is measured in Hertz (Hz).

HALF STEP Each key on the piano is a half step apart. The white keys that have no black key between them (E-F and B-C) are also a half step apart.

KEY The tonal center of a series of notes.

KEY SIGNATURE The sharps or flats placed at the beginning of a composition after the clef sign indicating the key of the music to follow.

MEASURE The musical interval that marks a repeat of the rhythm. Each measure is marked by a bar (vertical) line through the staff.

MELODY The chief or lead part of a harmonious composition.

METRONOME An instrument that can be adjusted to mark time exactly by a regularly repeated tick.

MIDDLE C The note that occupies a short line between the base clef and treble clef.

MUSIC The art of organizing sounds in order.

NOISE A complex sound without pure tones.

NOTE A written symbol for a unit of music that indicates pitch by its position on the staff and duration by its shape.

OVERTONE One of the higher notes that is produced by the resonance of a note. The natural division of music into notes is based on overtones.

PHRASE A short musical section usually two to four measures long.

PITCH The frequency of a specific note.

RESONANCE Vibration produced by stimulating an object with an external sound.

REST A rhythmic silence in the music.

RHYTHM The recurring pattern that makes up the forward movement of the music.

ROUND A continuously repeated melody in which different voices are singing different parts of the melody at the same time.

SIGNATURE Symbols and numbers at the beginning of written music that indicate the clef and time.

SOUND Vibrations perceived by the sense of hearing.

STAFF The horizontal lines and spaces on which music is written.

TIMBRE The special quality of a note created by its overtones.

TIME The measure of musical periods.

TIME SIGNATURE A fractional sign placed just after the clef sign and key signature. The lower number indicates the kind of note that gets one count in a measure, and the upper number indicates the number of counts in a measure. 3/4 time means that there are three quarter notes per measure (waltz time), with the quarter note getting one count.

TONE A musical sound of a specific quality, either a specific pitch or the overall sound of an instrument. A pure tone is a note with a single precise frequency.

TREBLE CLEF The staff of notes above middle C.

TUNE To adjust the pitch of a musical instrument to match another standard instrument or harmonize properly with other notes on the same instrument.

VIBRATION A steadily repeating movement that travels in waves through a conductor. Sound vibrations are transmitted to the ear by tiny back and forth movement of air molecules.

A BRIEF GLOSSARY OF BASIC COMMANDS

This short list only discusses terms from BASIC as they are used in the music programs, and is not intended to replace a good instruction manual on BASIC.

ABS Instructs the computer to ignore a minus sign if it occurs, thus treating all numbers following ABS as positive.

ARRAY Organized lists of information that can be recalled at any time by an array variable. Not a BASIC command.

ATASCII Atari's version of ASCII that contains graphic symbols unique to the Atari.

ASCII American Standard Code For Information Interchange. A code of zeros and ones that computers use to process and transmit text.

BASIC Beginners All-purpose Symbolic Instruction Code. A computer language that allows the user to operate the computer with almost understandable words.

CATALOG The list of programs on a disk. Not a BASIC command. Only available on the DOS menu.

DATA A command that indicates that what follows is information for the READ command. Atari sound values for the music are stored as DATA statements.

DOS Disk Operating System. Typing DOS <RETURN> loads a menu of helpful programs to use with the disk drive.

END A command that stops the operation of the program, turns off the sound, and leaves the computer ready for more instructions from the user.

ENTER Read a text file from a disk or tape. When you are editing programs, the parts are saved as text files with the LIST command and retrieved with ENTER.

GET Assign the value of the next key pressed to the variable that follows GET.

GOSUB Branch to the line number that follows the GOSUB command. Perform the steps that follow until a RETURN command is found. RETURN sends control back to the next statement after the GOSUB.

GOTO Program flow branches to the line number that follows GOTO.

LIST Write out all or part of the program lines to the screen, to a disk drive, or to the program recorder.

LOAD Get a program from disk drive or program recorder and load it into computer memory. LOAD erases previous programs and other information presently stored in the computer's memory. Programs can be combined with the LIST and ENTER commands.

MENU A list of choices on the computer screen. Not a BASIC command; you have to write menus into the program.

ON/GOSUB The ON command counts through a list of line numbers that follow the GOSUB command and instructs the computer to go to the line number indicated, do the commands that it finds there until a RETURN is found, at which point, the computer RETURNs to the next statement following the GOSUB.

OPEN Assigns a channel of input/output to the number that follows OPEN.

PEEK Get the value stored in the location indicated by the number that follows the PEEK command.

POKE Store a value in a specific memory location.

PROGRAM BASIC commands organized by line numbers to carry out a function when it is RUN.

READ Command to get the next item from the DATA statement and assign the item to the variable that follows the READ command. We use READ to get notes and time values for the music programs.

ROUTINE A section of a program that performs a special function.

REM Tells the computer that what follows is a message to humans, not a computer command.

RUN Runs the program currently in the computer, or gets a specified program from the disk drive and runs it.

SAVE Sends the current program to the disk drive or program recorder to be stored.

SOUND Atari's command that allows the computer to produce a sound. Each value in the command tells the computer something it needs to know to make sound:

1. Voice
2. Note value
3. Distortion
4. Loudness

TEXT All the symbols of the keyboard, plus a few more special symbols, stored in a standard format called ASCII.

VARIABLE A letter or letter/number combination that can be assigned to hold information.

NOTE: In the programs, we have given our variables names that are descriptive of their function, such as the ones listed below. These are not BASIC commands.

KEY FADE NUM PAUSE PICK PLAY TONE WAIT

Any of these words can be replaced with a single letter or with another word except one that contains a BASIC command such as RUNNER (starts with RUN) or ABSTRACT (starts with ABS) etc.

THE MUSICAL ATARI is an introduction to both music and computing. Dozens of popular songs are specially arranged so they can be played on the piano or the Atari with equal ease.

Written for the beginner, THE MUSICAL ATARI includes simple BASIC programs that allow you to play harmonies, melodies and rounds — converting any Atari Home Computer into a player piano or chord organ at the touch of a key.

The programs, music theory and notation are clearly explained, allowing non-musicians to play music written for piano, voice, or guitar on the Atari. A special chapter on sound effects includes short sound programs that can be incorporated into your own games. Also, the principles of BASIC programming covered in this fantastic book will help you develop games and other programs.

You don't need to be a musician to play beautiful music on the Atari, you only need THE MUSICAL ATARI!

ISBN 0-88190-345-0

DATAMOST
20660 Nordhoff Street, Chatsworth, CA 91311-6152
(818) 709-1202