Approximating Omega

(Originally formatted for 11 x 8.5 inch paper.
Magnify and print on larger paper such as 17 x 11
inch if document is too small to comfortably read)

-michael winter (la; 2010)

Approximating Omega

Note (to be used as an accompanying document for performances providing resources related to this piece):

In a 1975 paper entitled "A theory of program size complexity formally identical to information theory," Gregory Chaitin formally defines Omega, which is the probability that a computer program with bits generated by tosses of a fair coin halts on a universal self-delimiting Turing machine. Omega is maximally complex and incomputable because no algorithm can determine whether an arbitrary computer program halts. The undecidability of what is now known as the "halting problem" was originally shown by Alan Turing in his 1936 paper, "On computable numbers, with an application to the Entscheidungsproblem," as a corollary to a computer theoretical proof of Kurt Godel's incompleteness theorem first presented in Godel's 1931 article, "On formally undecidable propositions of Principia Mathematica and related systems I."

While Omega is incomputable, Chaitin has written a computer program that approximates Omega with increasing accuracy over longer and longer amounts of time. The program is written in a version of LISP that Chaitin extended and altered from the original LISP protocol first created by James MCarthy 1958 with a published explanation in his 1960 paper, "Recursive functions of symbolic expressions and their computation by machine."

The first [optional] section of this piece consists of an accompanied speaker reading an explanation of Chaitin's dialect of LISP. The text is adapted from Chaitin's 1994 book, "The Limits of Mathematics." Chaitin has further extended his 1994 LISP in his 2001 book, "Exploring Randomness." The decision to use Chaitin's older version of LISP and make changes to the text of "The Limits of Mathematics" are for artistic purposes with permission from the author.

The second section realizes symbol-by-symbol the program that approximates Omega. Distinct (primarily short) sounds represent each symbol and the entrances and exits of various continuous sounds demarcate the expressions in the program. Thus, the nesting of subroutines within the program completely determines the form of the music.

-michael winter (la; 2010)

Performance Instructions:

Dynamics and Setting:
All individual instruments should sound at a uniform dynamic; each sound heard clearly without being loud. Sound sources should be distributed throughout the performance space. Realizations should be primarily concerned with the phenomenon of sound itself.

Instrumentation:
Part 0 comprises a battery of 26 different sounds indicated by number. 0 and 1 indicate pitched sounds (pitch class – D; any fixed octave) from a percussion instrument with long natural decay. 0 indicates to let ring. 1 indicates a punctuated tone (quickly stopped). Sounds 0 and 1 occur most frequently. When possible, distribute these occurrences to a set of a particular instrument (such as 17 chimes of the same pitch located throughout the performance space) such that: when several 0s occur in succession, each sound comes from a different source; when a 0 is followed shortly by a 1, the same source produces both sounds (that is, the tone initially allowed to ring is stopped by the punctuated tone that follows). Each remaining sound-number in Part 0 indicates a distinct, non-pitched sound that is short and/or has a short natural decay. Distribute the sources of these sounds throughout the performance space to the extent possible. Each part from 1 through 17 must be distinguished uniquely by timbre and/or pitch. One performer may play several parts with double-stops, multiple instruments, polyphonic synthesized or recorded sounds, etc.. Assign each part played by a pitched instrument a distinct pitch derived from one of the first seventeen primes of the harmonic series of D. Note names with cent deviations (one hundredth of a tempered semitone) are provided above the charts in Section 2. Performers may sound a part's respective pitch-class in any octave and may change octaves for each new tone.

Section 1 (optional):
The text is read in a paced speaking voice with silences of varying lengths between each paragraph (delimited by double line breaks). Varying subsets of the ensemble accompany each paragraph starting and ending precisely with the speaker. Per paragraph, the accompanying sound should seem as a single gestalt. Sounds should generally by continuous however percussion instruments may choose to sound a series of punctuated attacks repeated quickly at individual tempi. Individual instruments may enter and exit freely so long as an overall continuity remains. Symbols bound by brackets are not spoken. Instead, a tone corresponding to the sound-number of Part 0 (given in the left column) sounds. Precede and succeed this tone with silence. The paragraph is interpreted to start on the following line. A word in quotes coincides with the sounding of a tone corresponding to the sound-number of Part 0 (also given in the left column). Underlined words are neither spoken nor accompanied. Performers of Part 0 must play when indicated by brackets or quotes in the text but may also contribute to the more continuous accompanying sound of each paragraph.

Section 2:
Part 0 is notated as a list of pairs. The first number is a time-unit within which a sound indicated by the second number occurs. Two charts provide options for the remaining 17 parts (organized by columns). The column contains a list of time-unit pairs for the start and end times of each tone. The tones must enter and exist precisely with the sound in Part 0 occurring in the same time-unit. The ensemble may uniformly scale the time-units by any amount. Note that in Option 1, the higher the part number, the shorter the general durations of tones. Option 2 sacrifices this for parts that are more uniformly active. More traditionally notated scores are also provided for both options. An appendix gives the list of time pairs without part assignment. The ensemble may explore distributing the tones in other ways but it will always take at least 17 parts.

Section 1:

Part 0:

Speaker with Accompaniment

Omega is formally defined as the probability that a computer program with bits generated by tosses of a fair coin halts on a universal self-delimiting computer. Since no algorithm can determine whether an arbitrary computer program halts, Omega is maximally complex and incomputable. However, there exists a computation that approximates Omega with increasing accuracy over longer and longer amounts of time. We outline this method in an altered and extended version of the computer programming language, LISP.

LISP more closely resembles fundamental subjects such as set theory and logic than a traditional programming language. The LISP formalism consists of several primitive functions and a set of rules for defining more complex functions from the initially given primitives. LISP functions are technically known as partial recursive functions. Data and function definitions in LISP consist of S-expressions. S stands for symbolic.

S-expressions are lists consisting of start and end delimiters binding zero or more elements, which may be atoms or sublists. Formally, the class of S-expressions is the union of the class of atoms and the class of lists.

The fundamental semantic concept of LISP is that of the value of an S-expression in a given environment. An environment consists of an associated list in which variables (atoms) and their values (S-expressions) alternate. If a variable appears several times, only its first value is significant. If a variable does not appear in the environment, then it is a literal constant in that it itself is its value.

LISP reserves the following symbols given with a name, the number of arguments (if applicable) and an explanation. All but the first four represent primitive functions.

0 Symbol: [(]
Name: Start-Delimiter
Explanation: Denotes the start of an S-expression.

1 Symbol: [)]
Name: End-Delimiter
Explanation: Denotes the end of an S-expression.

2 Symbol: [1]
Name: True
Explanation: Denotes the value true.

3 Symbol: [0]
Name: False
Explanation: Denotes the value false.

4 Symbol: [’]
Name: Quote or Literally
Arguments: 1
Explanation: The result of applying this function is the unevaluated argument expression.

5 Symbol: [.]
Name: Atom
Arguments: 1
Explanation: The result of applying this function to an argument is true or false depending on whether or not the argument is an atom.

6 Symbol: [=]
Name: Equal
Arguments: 2
Explanation: The result of applying this function is true or false depending on whether or not the two arguments are the same S-expression.

7 Symbol: [+]
Name: Head or First
Arguments: 1
Explanation: The result of applying this function to an atom is the atom itself. The result of applying this function to a non-empty list is the first element of the list.

8 Symbol: [-]
Name: Tail or Rest
Arguments: 1
Explanation: The result of applying this function to an atom is the atom itself. The result of applying this function to a non-empty list is the remaining elements after deletion of the first element. Thus, the tail of an (n + 1)-element list is an n-element list.

9 Symbol: [*]
Name: Join
Arguments: 2
Explanation: If the second argument is not a list, then the result of applying this function is the first argument. If the second argument is an n-element list, then the result of applying this function is the (n + 1)-element list whose head is the first argument and whose tail is the second argument.

10 Symbol: [,]
Name: Display
Arguments: 1
Explanation: The result of applying this function is its argument and is used to display intermediate results. In other words, this is an identity function. It is the only primitive function with a side-effect, which is to display the argument.

11 Symbol: [/]
Name: If-then-else
Arguments: 3
Explanation: If the first argument is not false, then the result is the second argument. If the first argument is false, then the result is the third argument. The argument that is not selected is not evaluated.

12 Symbol: [!]
Name: Evaluate
Arguments: 1
Explanation: The expression that is the value of the argument is evaluated in an empty environment. This is the only primitive function that is a partial rather than a total function.

13 Symbol: [?]
Name: Try or Depth-Limited Evaluation
Arguments: 2
Explanation: The expression that is the value of the second argument is evaluated in an empty environment. The number of elements of the first argument gives a time limit (that is, a maximum number of computations equal to the length of the list or zero if the first argument is not a list). The time limit actually limits the depth of the evaluation. If the evaluation completes within the time limit, the value returned is a list whose sole element is the value of the expression that is the value of the second argument. If the evaluation is not completed within the time limit, the value returned is the atom for “Try.”

14 Symbol: [&]
Name: Define Function or Lambda
Arguments: 2
Explanation: Treated essentially as a primitive function, this atom is used to create a defined function where the first argument is a list of variables and the second argument is the body of the function definition. Note that all other (non-reserved) symbols may be used as variables in a defined function.

We extend LISP to define a self-delimiting universal computer. The computer's program appears on its tape as a binary representation of a LISP expression. Note that the program must be self-delimiting because the S-expression must have balanced delimiters.

13 We redefine “Try” by adding an argument to be able to initially place information on the computer's tape. The three arguments are as follows. The first argument, the depth-limit, is altered from the original LISP definition: if it is a non-null atom, then there is no depth limit; if it is the empty list, there is zero depth limit (that is, no function calls or re-evaluations); if it is an n-element list, there is a depth limit of n. The second argument is as before: the expression to be evaluated as long as the depth limit is not exceeded. The new third argument is a list of bits to be used as the computer's program tape.

13 The value returned by “Try” is also changed. If the computation terminates normally, the first element of the returned value is a list with only one element, which is the result of the computation. If the evaluation of the second argument aborts, the first element of the
12 returned value is the atom for “Evaluate” after an attempt to read a non-existent bit from the
13 tape or the atom for “Try” when the number of computations exceeds the depth limit. The rest
10 of the returned value is a stack of all the arguments to the primitive function “Display”
encountered during the evaluation of the second argument.

We reserve two more symbols for primitives that could be programmed but are built-in to help conveniently define and efficiently run a self-delimiting universal computer.

15 Symbol: [^]
Name: Append
Arguments: 2
Explanation: The result of this function is the concatenation of its two arguments into a single list.

16 Symbol: [%]
Name: Read-Expression
Explanation: Read an entire LISP expression from the computer's tape. This function is the only way that information on the computer's tape can be accessed. It must be implemented in a self-delimiting fashion because no algorithm can search for the end of the tape and then use the length of the tape as data in the computation. If an algorithm attempts to read a bit that is not on the tape, the algorithm aborts. That is, this function explodes if the tape is exhausted, killing the computation.

In conclusion, permissiveness in our LISP is achieved because functions with extra arguments are evaluated but ignored and empty lists are supplied for missing arguments. There are no erroneous expressions; only expressions that never return a value because the interpreter goes into an infinite loop.

Approximating Omega:

Section 2:

```
(('(&(V)(('(&(A)(('(&(R)(('(&(W)(W('(O)))))('(&(n)(*0(*.(R(Vn())n))))))))('(&(xy)(/(.x)(/(.y)()(*0(Rx(-
y))))(^(R(-x)(y))(*(+x)())))))))))('(&(xyz)(/(.x)(/(.y)(/z('(1))())(A('(0))yz))(/(.y)(Ax('(0))z)(*(=(+x)
(=(+y)z))(A(-x)(-y)(/(+x)(/(+y)1z)(/(+y)z0)))))))))))('(&(xy)(/(.x)(/(.(+(?n('(!(%)))y)))()('(1)))(A(V(
-x)(*0y))(V(-x)(*1y))0))))))
```

Above is the program approximating Omega given in its ascii representation. Note that the symbol O represents a list of 1s with a length that determines how many bits of the binary expansion of Omega are approximated.

Part 0 (time-unit, sound):

0,	0	26,	4	52,	3	78,	22	104,	1	130,	1	156,	23	182,	0	208,	6	234,	0	260,	0	286,	12	312,	1
1,	0	27,	0	53,	0	79,	23	105,	1	131,	1	157,	1	183,	11	209,	0	235,	7	261,	4	287,	0	313,	0
2,	4	28,	14	54,	9	80,	1	106,	1	132,	1	158,	0	184,	0	210,	7	236,	23	262,	0	288,	16	314,	9
3,	0	29,	0	55,	5	81,	0	107,	1	133,	1	159,	11	185,	5	211,	23	237,	1	263,	14	289,	1	315,	3
4,	14	30,	20	56,	0	82,	11	108,	0	134,	1	160,	24	186,	23	212,	1	238,	2	264,	0	290,	1	316,	23
5,	0	31,	1	57,	19	83,	0	109,	15	135,	1	161,	0	187,	1	213,	24	239,	24	265,	22	291,	1	317,	1
6,	17	32,	0	58,	0	84,	5	110,	0	136,	1	162,	4	188,	0	214,	1	240,	1	266,	23	292,	23	318,	1
7,	1	33,	20	59,	17	85,	22	111,	19	137,	0	163,	0	189,	18	215,	1	241,	0	267,	1	293,	1	319,	0
8,	0	34,	0	60,	25	86,	1	112,	0	138,	4	164,	2	190,	22	216,	0	242,	11	268,	0	294,	1	320,	17
9,	0	35,	4	61,	0	87,	0	113,	8	139,	0	165,	1	191,	0	217,	18	243,	0	269,	11	295,	1	321,	0
10,	4	36,	0	62,	1	88,	11	114,	22	140,	14	166,	1	192,	4	218,	0	244,	7	270,	0	296,	0	322,	8
11,	0	37,	21	63,	1	89,	0	115,	1	141,	0	167,	0	193,	0	219,	8	245,	23	271,	5	297,	1	323,	22
12,	14	38,	1	64,	25	90,	5	116,	0	142,	22	168,	1	194,	3	220,	22	246,	1	272,	22	298,	0	324,	1
13,	0	39,	1	65,	1	91,	23	117,	8	143,	23	169,	1	195,	1	221,	1	247,	24	273,	1	299,	4	325,	0
14,	18	40,	1	66,	1	92,	1	118,	23	144,	24	170,	0	196,	1	222,	0	248,	3	274,	0	300,	0	326,	9
15,	1	41,	1	67,	1	93,	0	119,	1	145,	1	171,	18	197,	24	223,	8	249,	1	275,	11	301,	2	327,	2
16,	0	42,	1	68,	1	94,	1	120,	1	146,	0	172,	0	198,	1	224,	23	250,	1	276,	0	302,	1	328,	23
17,	0	43,	0	69,	1	95,	0	121,	0	147,	11	173,	4	199,	0	225,	1	251,	1	277,	5	303,	1	329,	1
18,	4	44,	4	70,	1	96,	9	122,	9	148,	0	174,	0	200,	9	226,	0	252,	1	278,	0	304,	1	330,	1
19,	0	45,	0	71,	1	97,	3	123,	0	149,	5	175,	3	201,	0	227,	11	253,	1	279,	7	305,	0	331,	3
20,	14	46,	14	72,	1	98,	0	124,	7	150,	22	176,	1	202,	6	228,	0	254,	1	280,	0	306,	18	332,	1
21,	0	47,	0	73,	0	99,	19	125,	22	151,	1	177,	1	203,	0	229,	7	255,	1	281,	13	307,	0	333,	1
22,	19	48,	25	74,	4	100,	22	126,	1	152,	0	178,	23	204,	7	230,	22	256,	1	282,	25	308,	17	334,	1
23,	1	49,	1	75,	0	101,	0	127,	0	153,	11	179,	24	205,	22	231,	1	257,	1	283,	0	309,	0	335,	1
24,	0	50,	0	76,	14	102,	8	128,	1	154,	0	180,	1	206,	1	232,	0	258,	1	284,	4	310,	8	336,	1
25,	0	51,	9	77,	0	103,	23	129,	1	155,	5	181,	1	207,	0	233,	11	259,	1	285,	0	311,	22		

Assign each part played by a pitched instrument a distinct pitch class from the following:
(D+0, A+2, F#-14, C-31, G#-49, A#+41, D#+5, F-2, G#+28, C+30, C#+45, F-49, F#+29, G+12, A-34, B-26, C#-41)

Option 1 (start time-unit, end-time unit):

Part 1	Part 2	Part 3	Part 4	Part 5	Part 6	Part 7	Part 8	Part 9	Part 10	Part 11	Part 12	Part 13	Part 14	Part 15	Part 16	Part 17
0,336	1,259	3,258	5,7	9,136	11,135	13,15	17,72	19,71	21,23	25,42	27,41	29,31	34,39	36,38	58,63	61,62
	260,335	262,334	8,257	137,256	139,255	16,134	73,133	75,132	24,70	43,69	45,68	32,40	53,66	56,65		
			264,267	270,273	276,295	141,145	148,151	154,157	77,80	83,86	89,92	47,49	101,104			
			268,333	274,304	296,297	146,254	152,181	158,169	81,131	87,107	93,94	50,67				
				305,332	298,303	278,294	182,253	170,180	161,166	108,130	95,106	98,105				
					307,318	300,302	280,293	184,187	167,168	163,165	110,120	112,115				
					319,330	309,312		188,198	172,177	174,176	121,129	116,119				
						313,317		199,252	191,196	193,195	209,212	123,126				
						321,324		283,291	201,215	203,206	228,231	127,128				
						325,329			216,251	207,214	232,240	234,237				
									285,290	218,221	241,249	243,246				
										222,225						
										226,250						
										287,289						

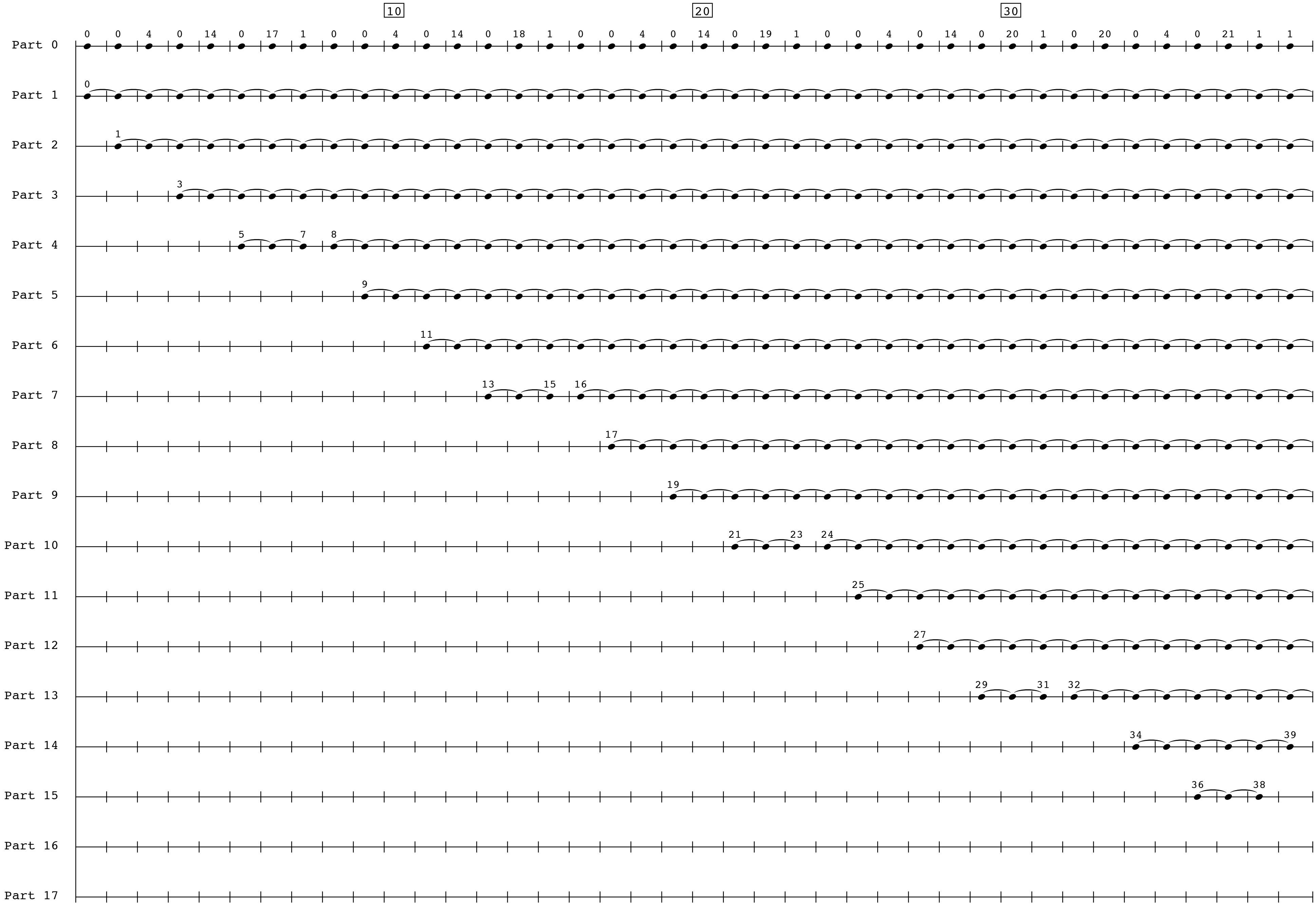
Option 2 (start time-unit, end-time unit):

Part 1	Part 2	Part 3	Part 4	Part 5	Part 6	Part 7	Part 8	Part 9	Part 10	Part 11	Part 12	Part 13	Part 14	Part 15	Part 16	Part 17
0,336	1,259	3,258	5,7	8,257	9,136	11,135	13,15	167,168	17,72	19,71	21,23	24,70	25,42	27,41	29,31	32,40
	305,332	300,302	34,39	298,303	172,177	170,180	36,38	193,195	98,105	95,106	43,69	93,94	58,63	56,65	45,68	53,66
			50,67		203,206	207,214	47,49	218,221	121,129	123,126	89,92	112,115	75,132	77,80	87,107	81,131
			83,86		228,231	234,237	61,62	243,246	152,181	146,254	110,120	137,256	161,166	101,104	127,128	158,169
			108,130		262,334	264,267	73,133	270,273	209,212	285,290	141,145	296,297	191,196	116,119	148,151	199,252
			154,157			307,318	163,165	309,312	232,240	319,330	174,176		222,225	139,255	182,253	280,293
			184,187				188,198		268,333		201,215		260,335	287,289	283,291	325,329
			216,251				226,250				241,249			313,317	321,324	
			278,294				276,295				274,304					

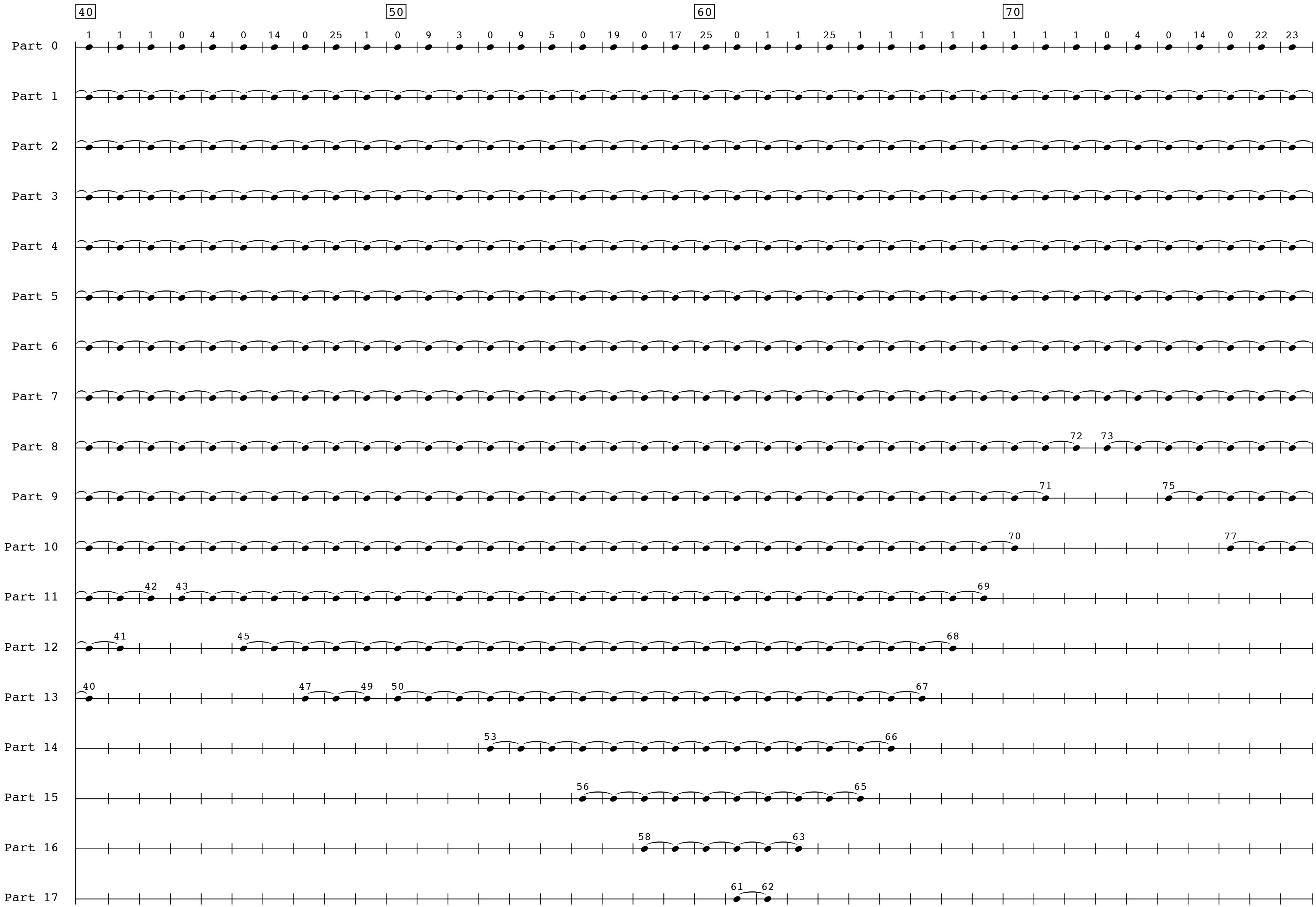
Appendix
Sustained Tones (start time-unit, end-time unit):
((0,336),(1,259),(3,258),(5,7),(8,257),(9,136),(11,135),(13,15),(16,134),(17,72),(19,71),(21,23),
(24,70),(25,42),(27,41),(29,31),(32,40),(34,39),(36,38),(43,69),(45,68),(47,49),(50,67),(53,66),
(56,65),(58,63),(61,62),(73,133),(75,132),(77,80),(81,131),(83,86),(87,107),(89,92),(93,94),(95,106),
(98,105),(101,104),(108,130),(110,120),(112,115),(116,119),(121,129),(123,126),(127,128),(137,256),
(139,255),(141,145),(146,254),(148,151),(152,181),(154,157),(158,169),(161,166),(163,165),(167,168),
(170,180),(172,177),(174,176),(182,253),(184,187),(188,198),(191,196),(193,195),(199,252),(201,215),
(203,206),(207,214),(209,212),(216,251),(218,221),(222,225),(226,250),(228,231),(232,240),(234,237),
(241,249),(243,246),(260,335),(262,334),(264,267),(268,333),(270,273),(274,304),(276,295),(278,294),
(280,293),(283,291),(285,290),(287,289),(296,297),(298,303),(300,302),(305,332),(307,318),(309,312),
(313,317),(319,330),(321,324),(325,329));

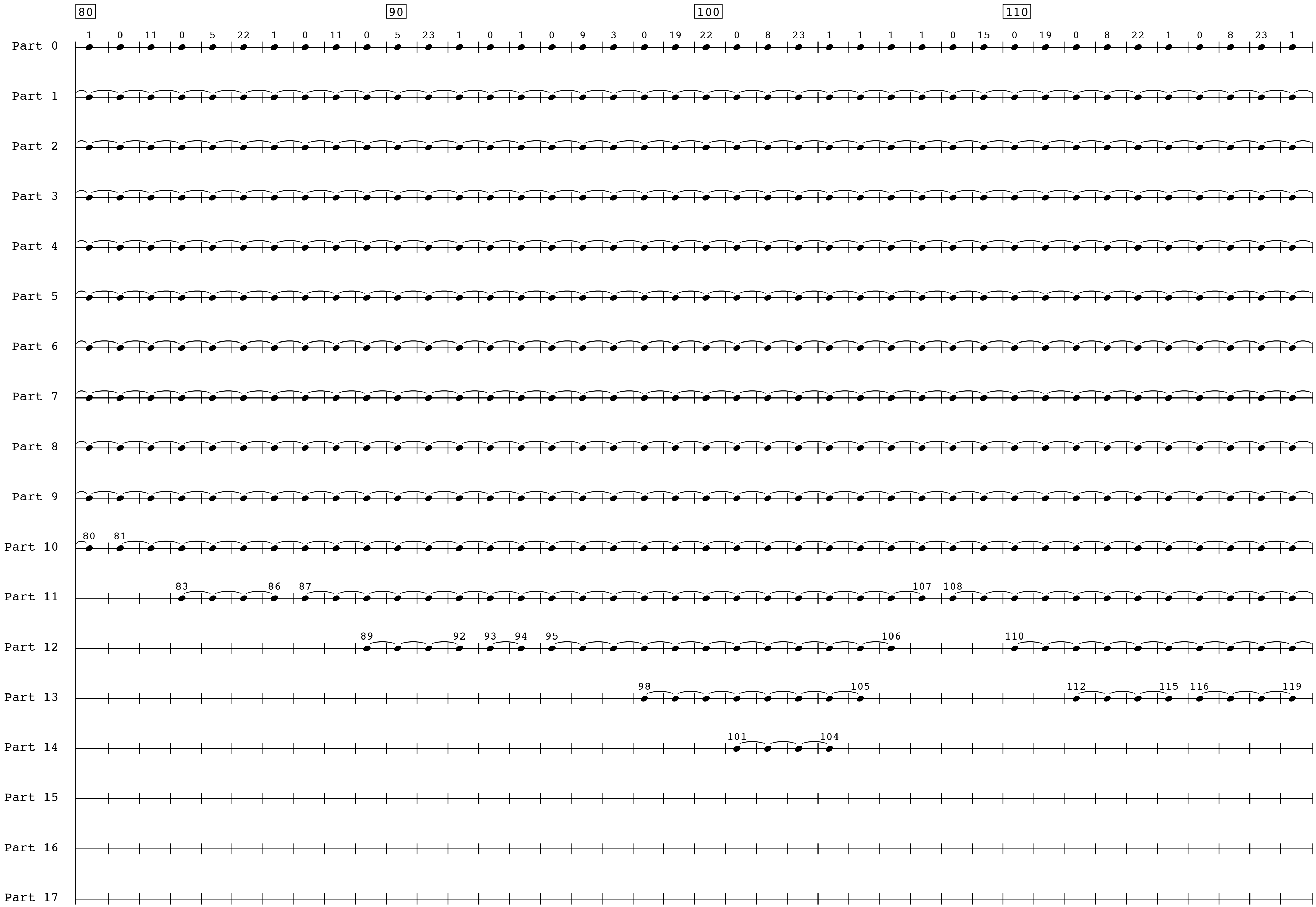
Approximating Omega
Section 2; Option 1

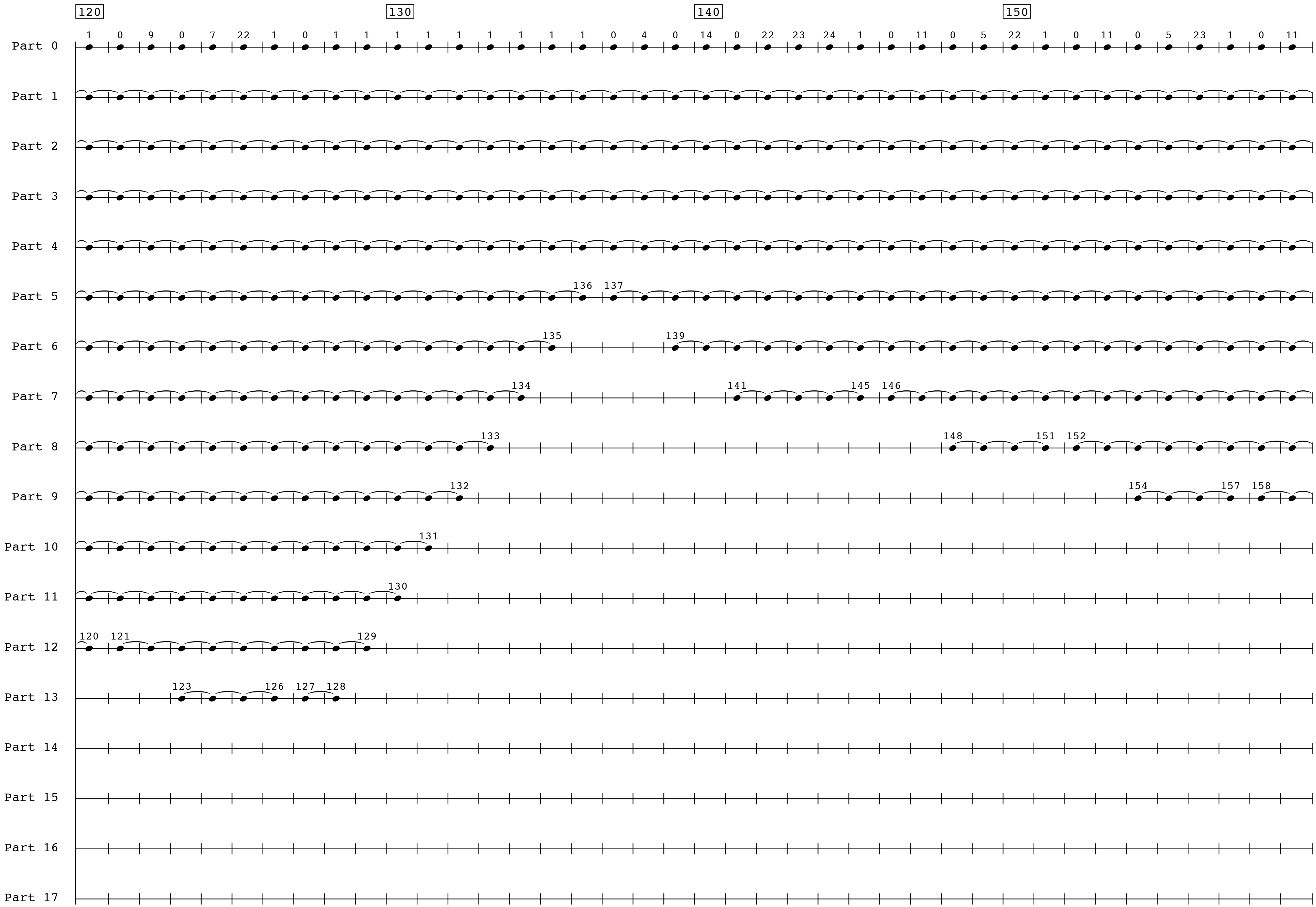
Assign each part from 1 through 17 played by a pitched instrument a distinct pitch class from the following (each tone may be played in any octave):
(D+0, A+2, F#-14, C-31, G#-49, A#+41, D#+5, F-2, G#+28, C+30, C#+45, F-49, F#+29, G+12, A-34, B-26, C#-41)

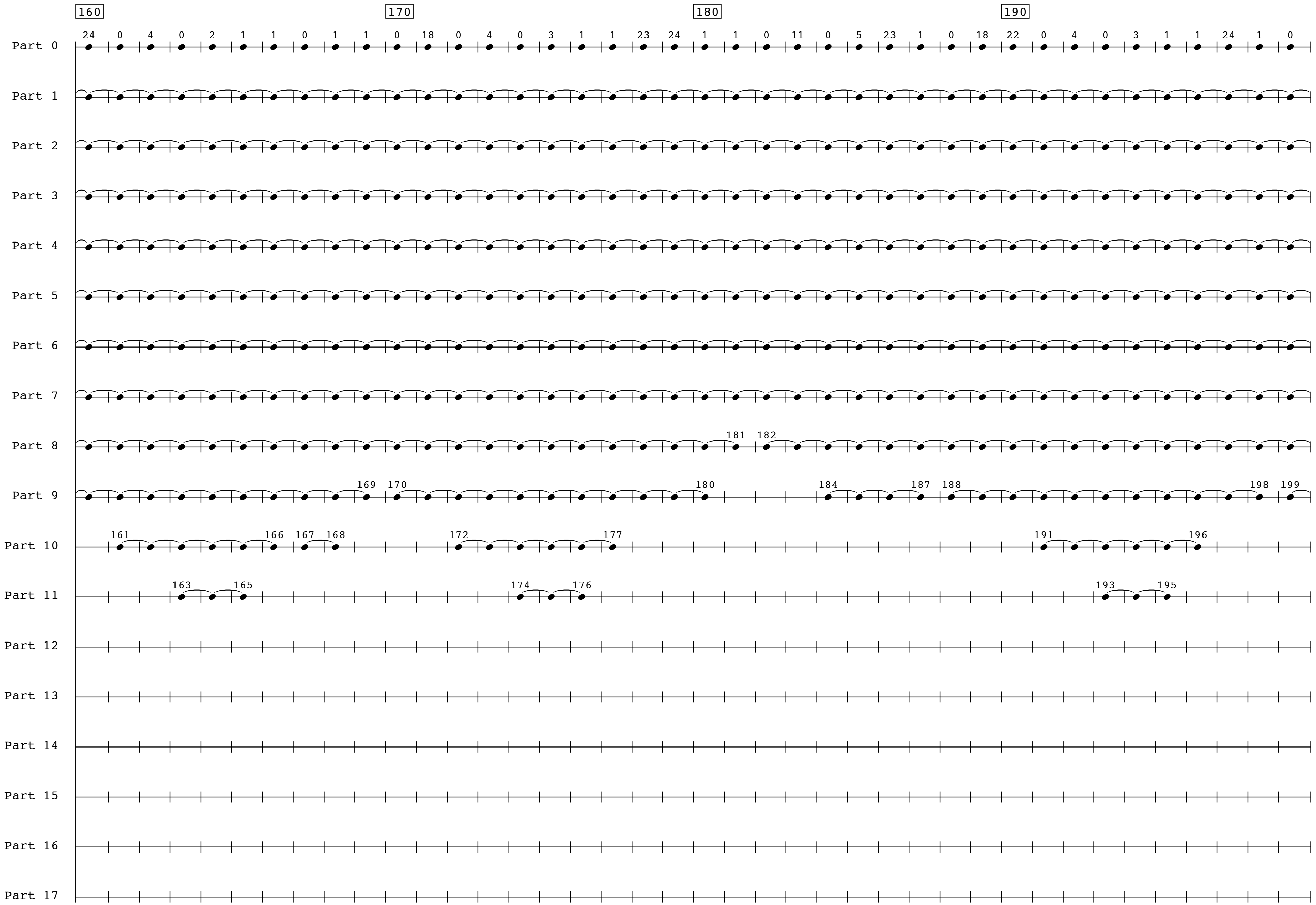


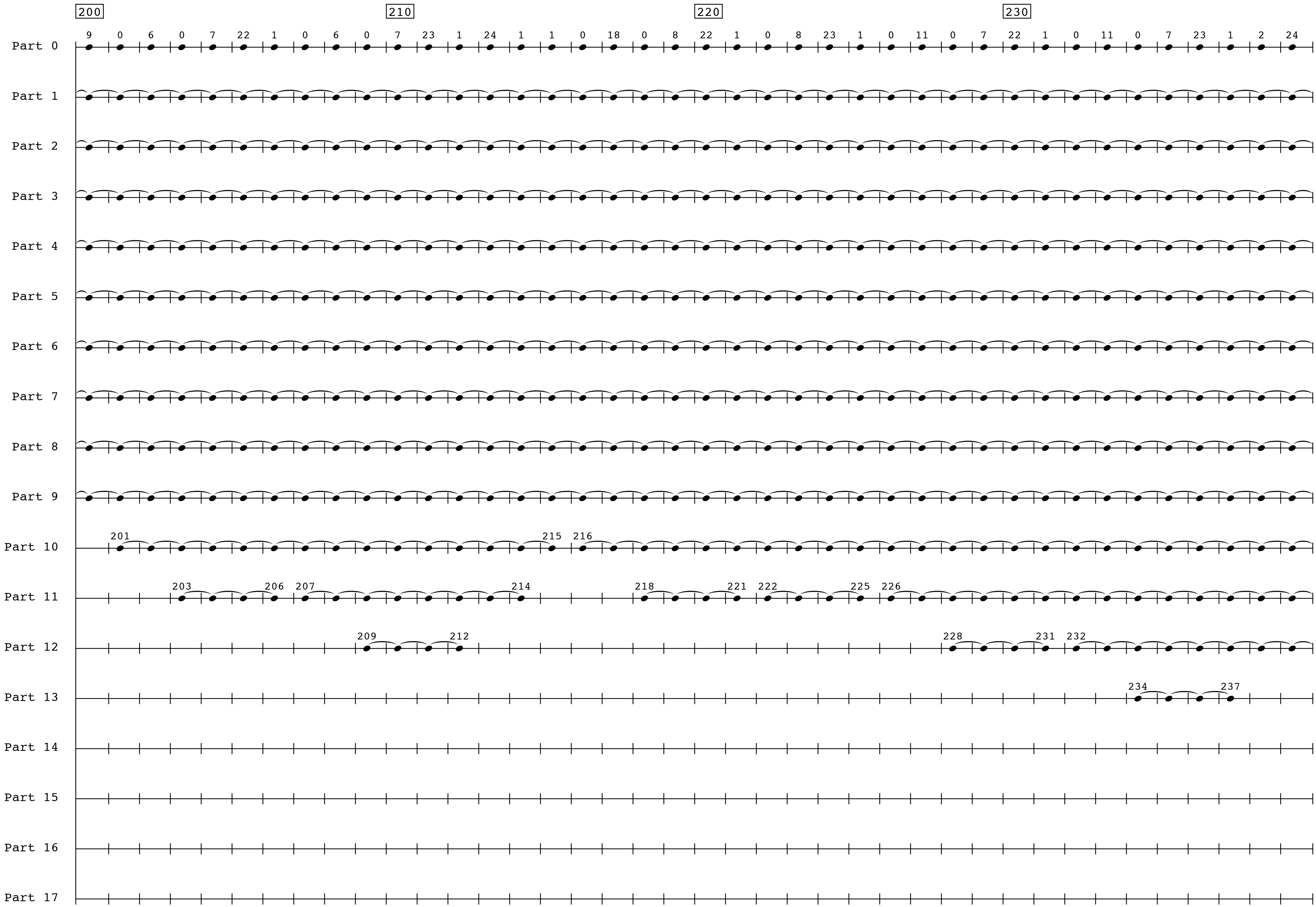
*Part 0: The number above each note indicates which sound to play. The sound may occur at any point in the time unit (or measure).
*Parts 1 through 17: The numbers above the starts and ends of each note indicate the time-unit (or measure) of entry or exit. These should coincide presicely with the sound occuring in part 1, which may occur at any point in the time-unit. Note that measure numbers start from index 0.

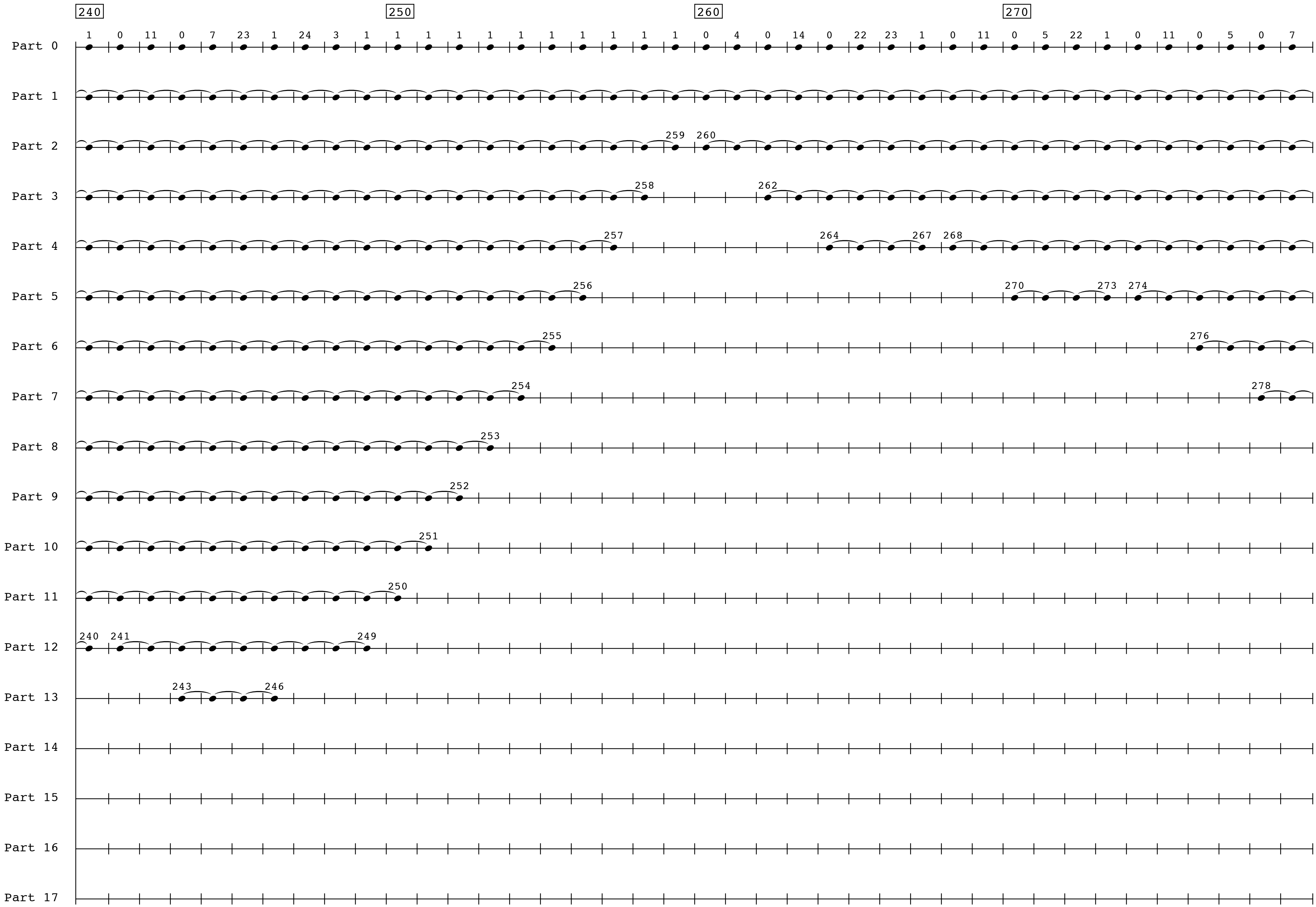

40
50
60
70
1 1 1 0 4 0 14 0 25 1 0 9 3 0 9 5 0 19 0 17 25 0 1 1 25 1 1 1 1 1 1 1 1 0 4 0 14 0 22 23
Part 0
Part 1
Part 2
Part 3
Part 4
Part 5
Part 6
Part 7
Part 8
72 73
Part 9
71 75
Part 10
70 77
Part 11
42 43 69
Part 12
41 45 68
Part 13
40 47 49 50 67
Part 14
53 66
Part 15
56 65
Part 16
58 63
Part 17
61 62

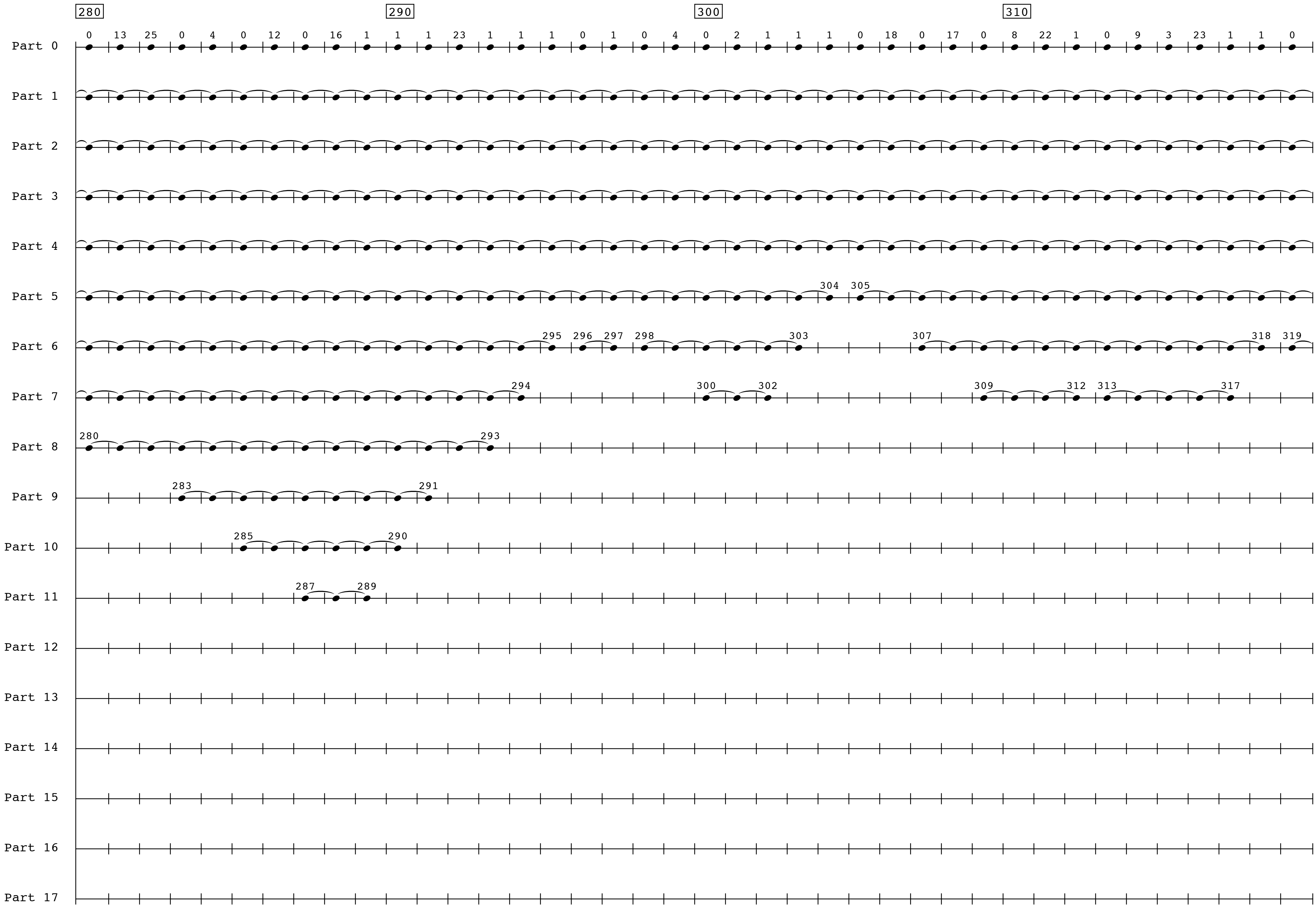

80
90
100
110
1 0 11 0 5 22 1 0 11 0 5 23 1 0 1 0 9 3 0 19 22 0 8 23 1 1 1 1 0 15 0 19 0 8 22 1 0 8 23 1
Part 0
Part 1
Part 2
Part 3
Part 4
Part 5
Part 6
Part 7
Part 8
Part 9
Part 10
80 81
Part 11
83 86 87 107 108
Part 12
89 92 93 94 95 106 110
Part 13
98 105 112 115 116 119
Part 14
101 104
Part 15
Part 16
Part 17

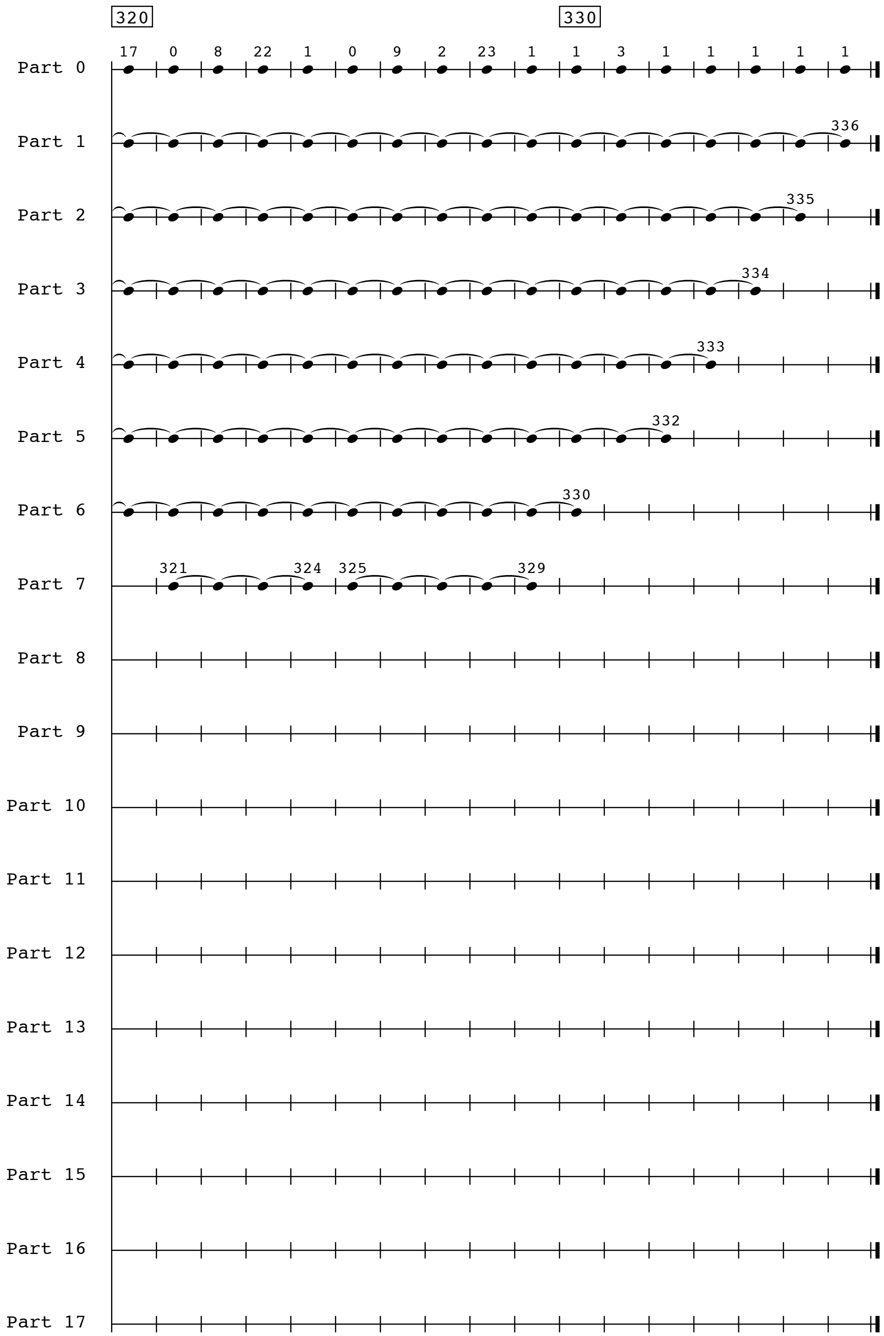

120
130
140
150
1 0 9 0 7 22 1 0 1 1 1 1 1 1 1 1 1 0 4 0 14 0 22 23 24 1 0 11 0 5 22 1 0 11 0 5 23 1 0 11
Part 0
Part 1
Part 2
Part 3
Part 4
Part 5
136 137
Part 6
135 139
Part 7
134 141 145 146
Part 8
133 148 151 152
Part 9
132 154 157 158
Part 10
131
Part 11
130
Part 12
120 121 129
Part 13
123 126 127 128
Part 14
Part 15
Part 16
Part 17


160
170
180
190
24 0 4 0 2 1 1 0 1 1 0 18 0 4 0 3 1 1 23 24 1 1 0 11 0 5 23 1 0 18 22 0 4 0 3 1 1 24 1 0
Part 0
Part 1
Part 2
Part 3
Part 4
Part 5
Part 6
Part 7
Part 8
181 182
Part 9
169 170 180 184 187 188 198 199
Part 10
161 166 167 168 172 177 191 196
Part 11
163 165 174 176 193 195
Part 12
Part 13
Part 14
Part 15
Part 16
Part 17


200
210
220
230
9 0 6 0 7 22 1 0 6 0 7 23 1 24 1 1 0 18 0 8 22 1 0 8 23 1 0 11 0 7 22 1 0 11 0 7 23 1 2 24
Part 0
Part 1
Part 2
Part 3
Part 4
Part 5
Part 6
Part 7
Part 8
Part 9
Part 10
201
215
216
Part 11
203
206
207
214
218
221
222
225
226
Part 12
209
212
228
231
232
Part 13
234
237
Part 14
Part 15
Part 16
Part 17


240
250
260
270
Part 0
1 0 11 0 7 23 1 24 3 1 1 1 1 1 1 1 1 1 1 1 0 4 0 14 0 22 23 1 0 11 0 5 22 1 0 11 0 5 0 7
Part 1
Part 2
259 260
Part 3
258 262
Part 4
257 264 267 268
Part 5
256 270 273 274
Part 6
255 276
Part 7
254 278
Part 8
253
Part 9
252
Part 10
251
Part 11
250
Part 12
240 241 249
Part 13
243 246
Part 14
Part 15
Part 16
Part 17

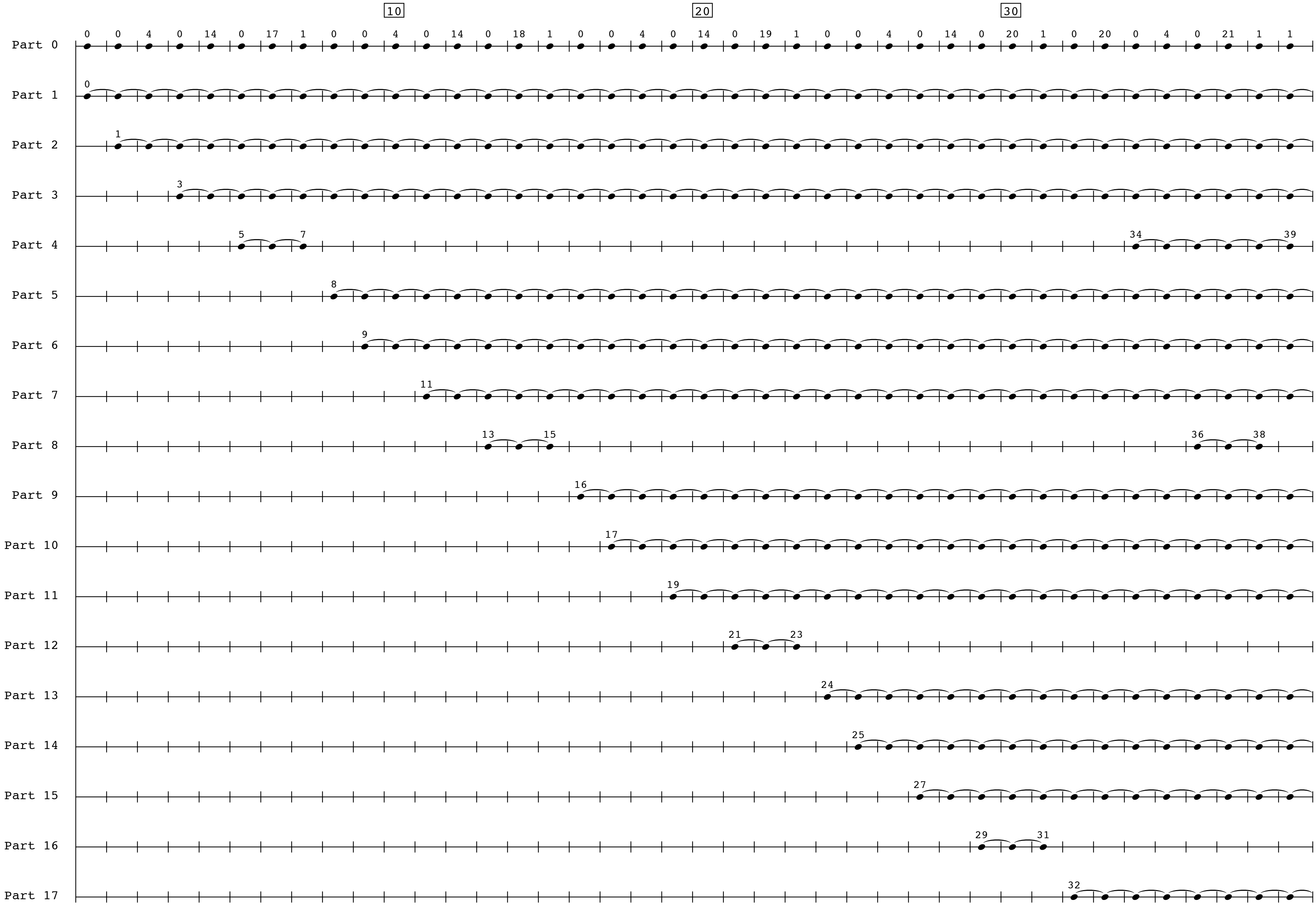

280
290
300
310
0 13 25 0 4 0 12 0 16 1 1 1 23 1 1 1 0 1 0 4 0 2 1 1 1 0 18 0 17 0 8 22 1 0 9 3 23 1 1 0
Part 0
Part 1
Part 2
Part 3
Part 4
Part 5
304 305
Part 6
295 296 297 298 303 307 318 319
Part 7
294 300 302 309 312 313 317
Part 8
280 293
Part 9
283 291
Part 10
285 290
Part 11
287 289
Part 12
Part 13
Part 14
Part 15
Part 16
Part 17



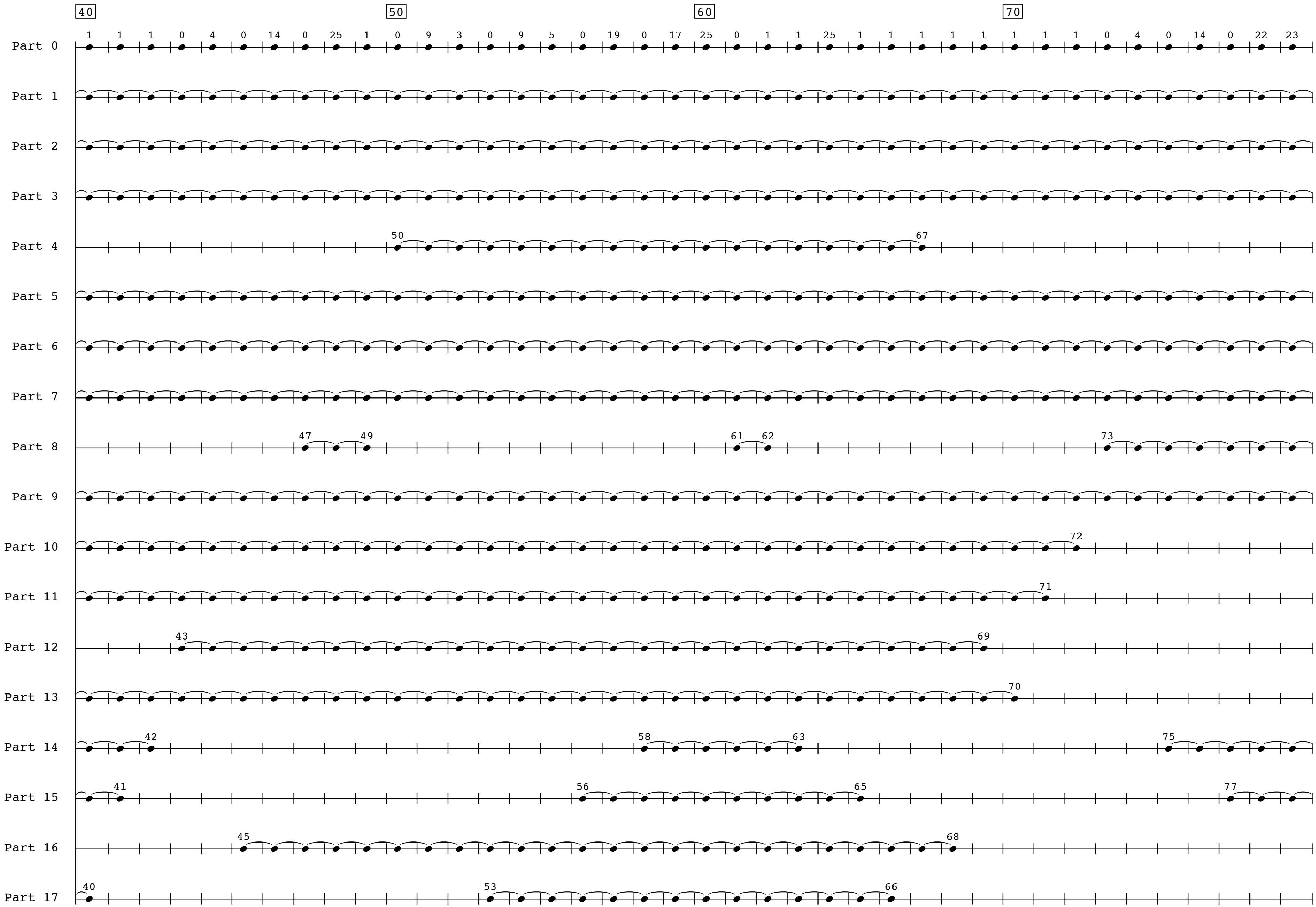
-michael winter (la; 2010)

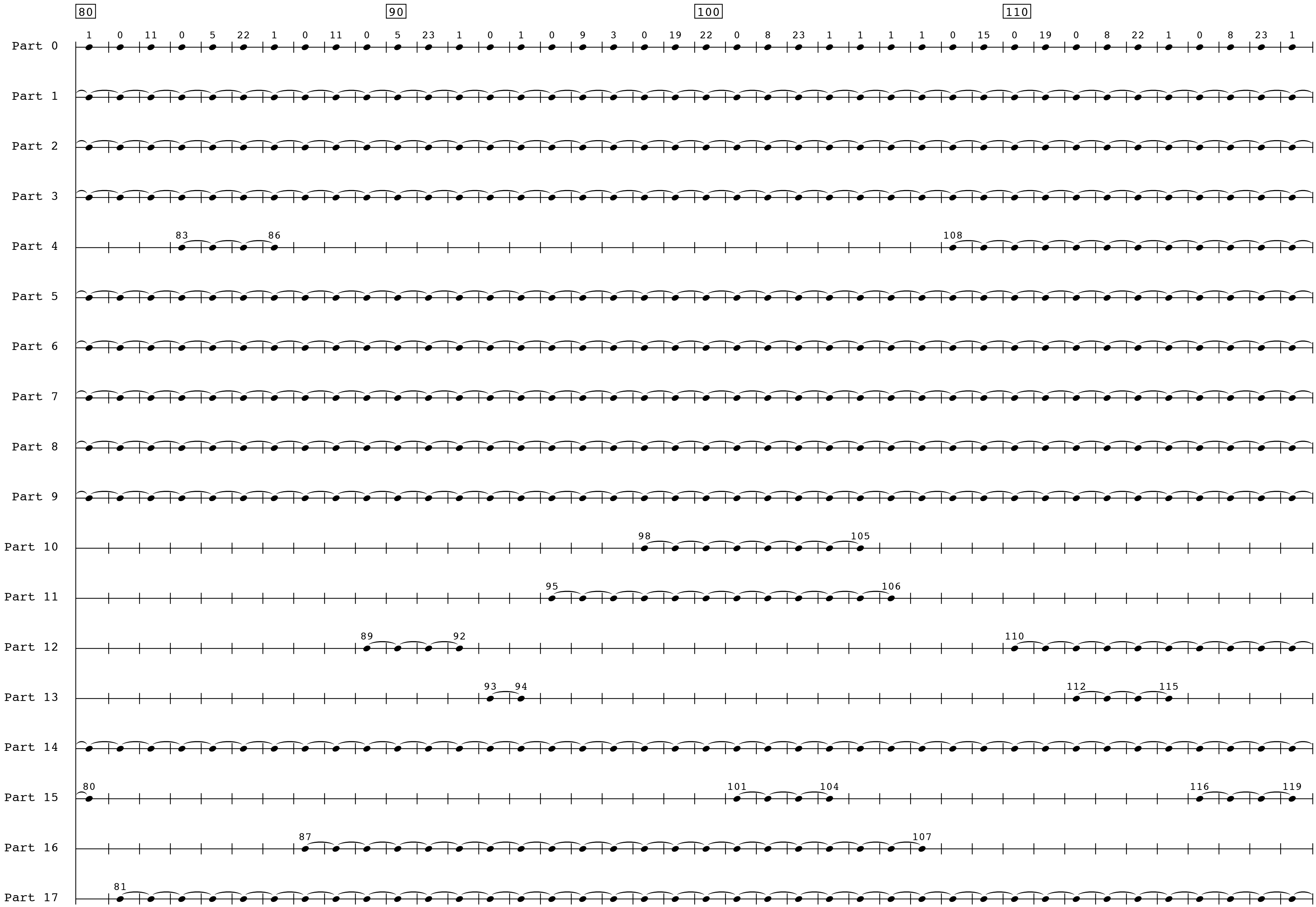
Approximating Omega
Section 2; Option 2

*Assign each part from 1 through 17 played by a pitched instrument a distinct pitch-class from the following (each tone may be played in any octave):
(D+0, A+2, F#-14, C-31, G#-49, A#+41, D#+5, F-2, G#+28, C+30, C#+45, F-49, F#+29, G+12, A-34, B-26, C#-41)

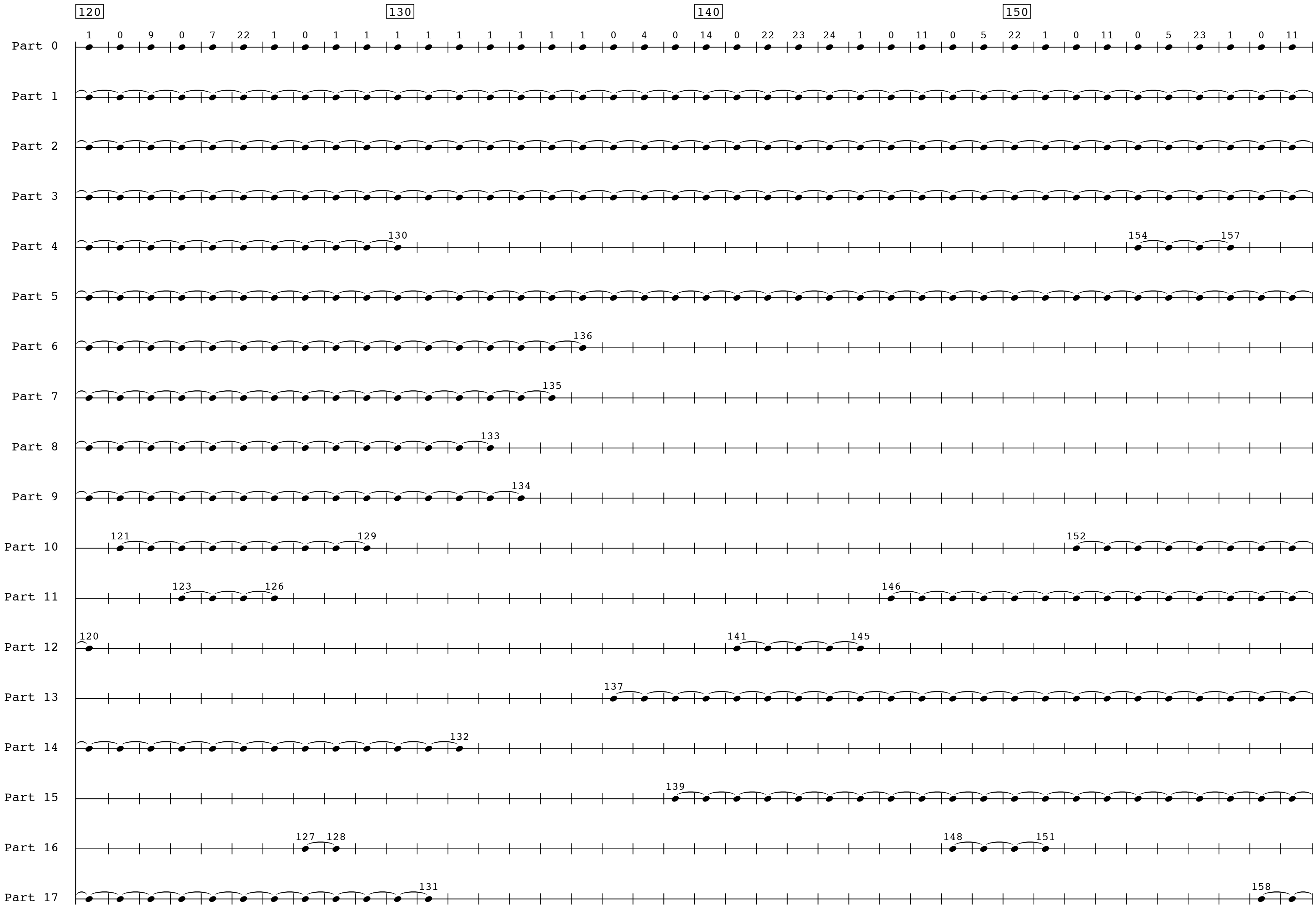


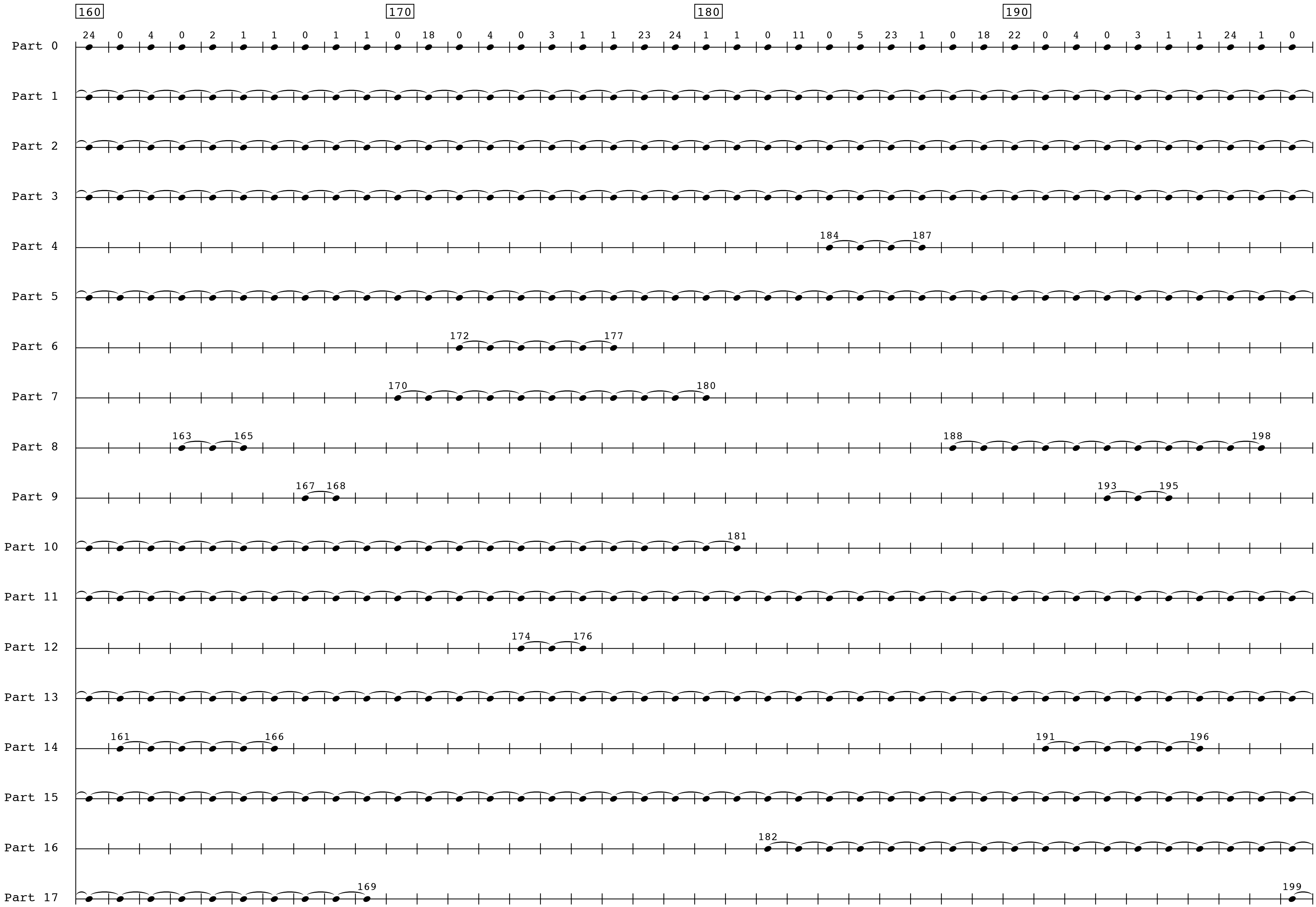
*Part 0: The number above each note indicates which sound to play. The sound may occur at any point in the time unit (or measure).
*Parts 1 through 17: The numbers above the starts and ends of each note indicate the time-unit (or measure) of entry or exit. These should coincide presicely with the sound occuring in part 1, which may occur at any point in the time-unit. Note that measure numbers start from index 0.

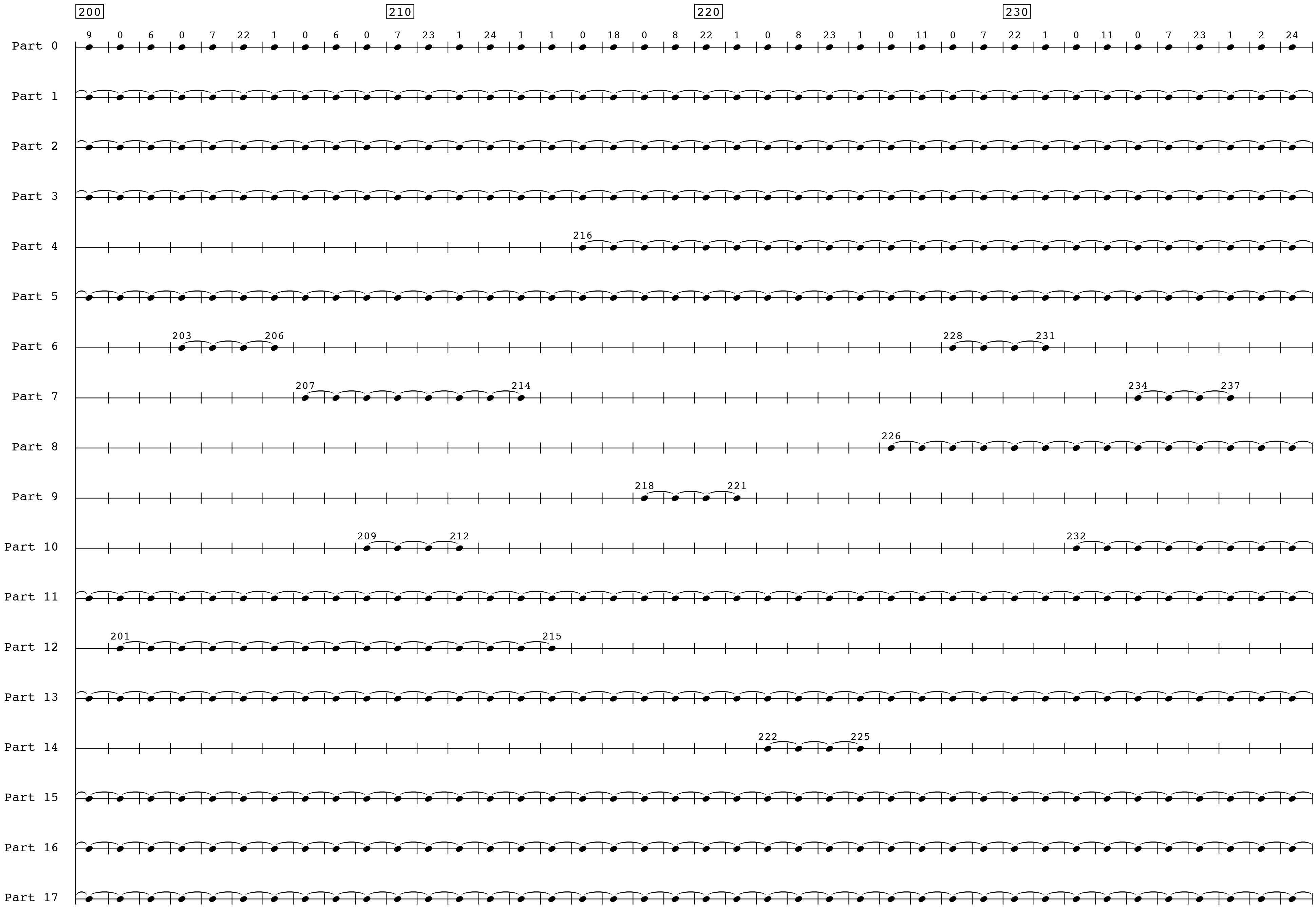

40
50
60
70
1 1 1 0 4 0 14 0 25 1 0 9 3 0 9 5 0 19 0 17 25 0 1 1 25 1 1 1 1 1 1 1 1 0 4 0 14 0 22 23
Part 0
Part 1
Part 2
Part 3
Part 4
50
67
Part 5
Part 6
Part 7
Part 8
47
49
61
62
73
Part 9
Part 10
72
Part 11
71
Part 12
43
69
Part 13
70
Part 14
42
58
63
75
Part 15
41
56
65
77
Part 16
45
68
Part 17
40
53
66

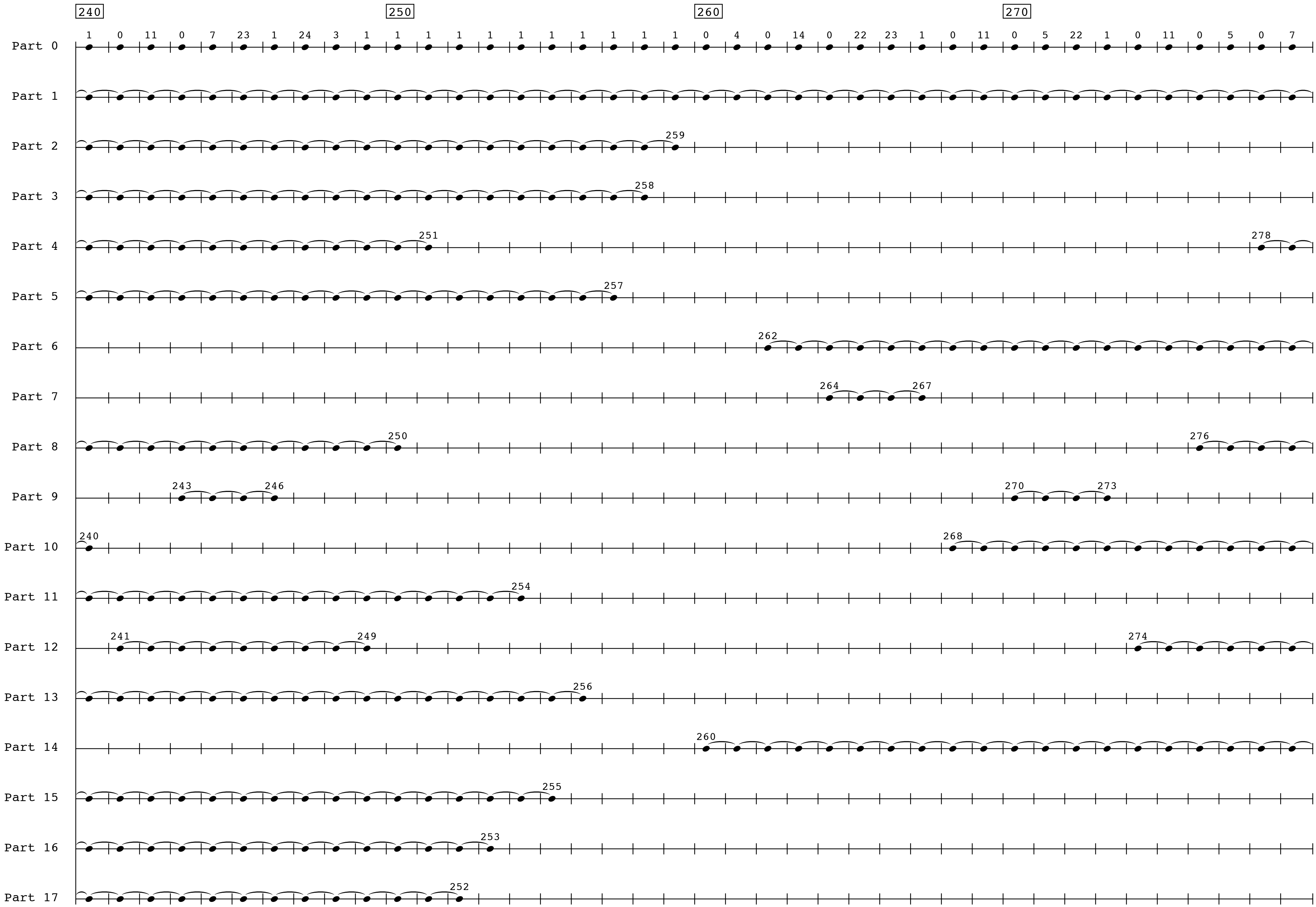


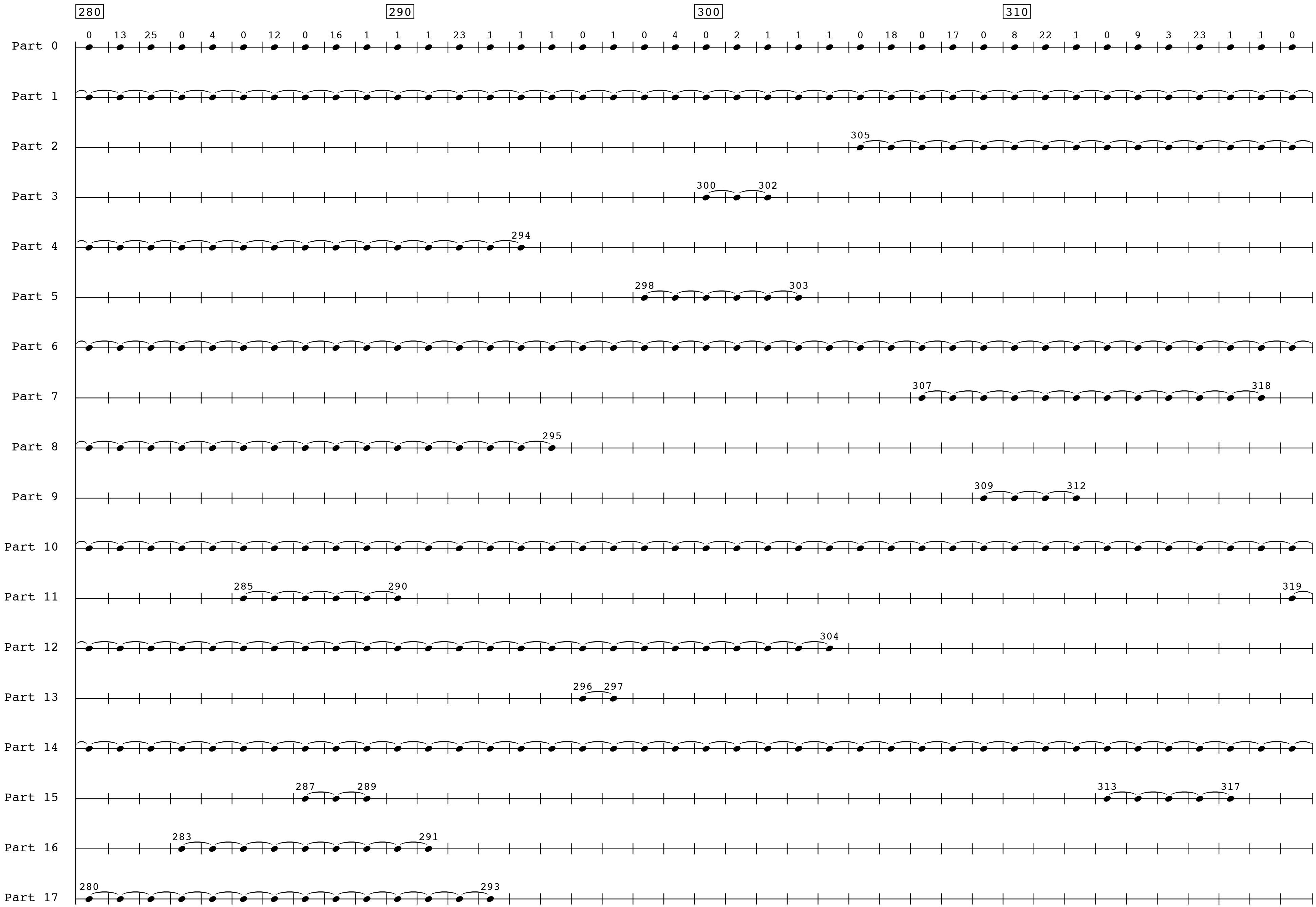
80
90
100
110
1 0 11 0 5 22 1 0 11 0 5 23 1 0 1 0 9 3 0 19 22 0 8 23 1 1 1 1 0 15 0 19 0 8 22 1 0 8 23 1
Part 0
Part 1
Part 2
Part 3
Part 4
83
86
108
Part 5
Part 6
Part 7
Part 8
Part 9
Part 10
98
105
Part 11
95
106
Part 12
89
92
110
Part 13
93
94
112
115
Part 14
Part 15
80
101
104
116
119
Part 16
87
107
Part 17
81

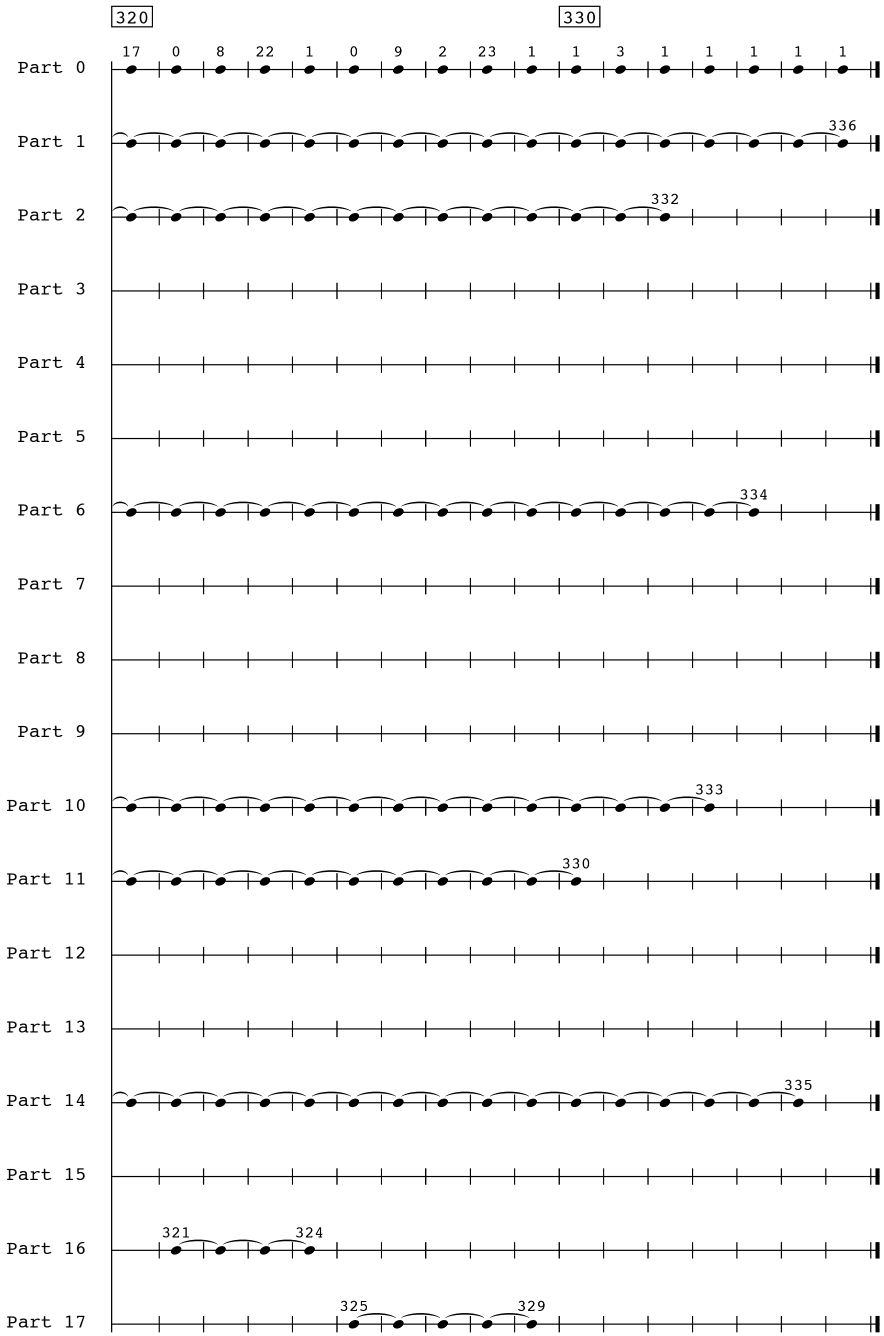

120
130
140
150
1 0 9 0 7 22 1 0 1 1 1 1 1 1 1 1 1 0 4 0 14 0 22 23 24 1 0 11 0 5 22 1 0 11 0 5 23 1 0 11
Part 0
Part 1
Part 2
Part 3
Part 4
130
154
157
Part 5
Part 6
136
Part 7
135
Part 8
133
Part 9
134
Part 10
121
129
152
Part 11
123
126
146
Part 12
120
141
145
Part 13
137
Part 14
132
Part 15
139
Part 16
127
128
148
151
Part 17
131
158


160
170
180
190
24 0 4 0 2 1 1 0 1 1 0 18 0 4 0 3 1 1 23 24 1 1 0 11 0 5 23 1 0 18 22 0 4 0 3 1 1 24 1 0
Part 0
Part 1
Part 2
Part 3
Part 4
184
187
Part 5
Part 6
172
177
Part 7
170
180
Part 8
163
165
188
198
Part 9
167
168
193
195
Part 10
181
Part 11
Part 12
174
176
Part 13
Part 14
161
166
191
196
Part 15
Part 16
182
Part 17
169
199


200
210
220
230
9 0 6 0 7 22 1 0 6 0 7 23 1 24 1 1 0 18 0 8 22 1 0 8 23 1 0 11 0 7 22 1 0 11 0 7 23 1 2 24
Part 0
Part 1
Part 2
Part 3
Part 4
216
Part 5
Part 6
203
206
228
231
Part 7
207
214
234
237
Part 8
226
Part 9
218
221
Part 10
209
212
232
Part 11
Part 12
201
215
Part 13
Part 14
222
225
Part 15
Part 16
Part 17


240
250
260
270
1 0 11 0 7 23 1 24 3 1 1 1 1 1 1 1 1 1 1 1 0 4 0 14 0 22 23 1 0 11 0 5 22 1 0 11 0 5 0 7
Part 0
Part 1
Part 2
259
Part 3
258
Part 4
251
278
Part 5
257
Part 6
262
Part 7
264
267
Part 8
250
276
Part 9
243
246
270
273
Part 10
240
268
Part 11
254
Part 12
241
249
274
Part 13
256
Part 14
260
Part 15
255
Part 16
253
Part 17
252


280
290
300
310
0 13 25 0 4 0 12 0 16 1 1 1 23 1 1 1 0 1 0 4 0 2 1 1 1 0 18 0 17 0 8 22 1 0 9 3 23 1 1 0
Part 0
Part 1
Part 2
305
Part 3
300
302
Part 4
294
Part 5
298
303
Part 6
Part 7
307
318
Part 8
295
Part 9
309
312
Part 10
Part 11
285
290
319
Part 12
304
Part 13
296
297
Part 14
Part 15
287
289
313
317
Part 16
283
291
Part 17
280
293



-michael winter (la; 2010)