

B. Subject Workbook

Introduction

In this experiment we hope to discover whether a technique known as musical program auralisation creates musical representations of Pascal programs that are easily recognisable. To do this experiment you do not need to be especially musical. The test is trying to determine how the average person makes use of musical information. We hope to have people with many different levels of musical ability taking part. Some will be highly trained musicians. Others will describe themselves as tone deaf with no interest in music. We need all types for our study, so do not worry if you consider yourself not to be musical.

You cannot pass or fail this test. We are simply interested in how recognisable you find the musical program auralisations. You do not need to be an expert programmer to take part and we are not assessing your ability as programmers. This study is completely separate from your degree course and the results will not be used in any way in regard to assessment or evaluation of your performance on your course of study.

Tutorial

CAITLIN is a tool that represents Pascal programs using musical tunes. The system takes a Turbo Pascal program and produces a version of that program in which all the constructs (loops & selections) are sounded by a musical tune played on a synthesiser during execution of the program.

For example, consider the following Pascal program:

```
PROGRAM Example ;
VAR
  counter,
  number      : Integer ;
BEGIN
  FOR counter := 1 TO 6 DO
    Writeln (counter) ;
  number := 10 ;
  IF number = 9 THEN
    Writeln ('Number is 9')
  ELSE
    Writeln ('Number is not 9') ;
END.
```

After processing by CAITLIN, the program would run as normal, but when the **FOR** loop is encountered then a musical tune representing the construct would be heard. Then, when the **IF** statement is executed, it too would be represented musically.

This process of representing the execution of a program using sound is known as **auralisation**. CAITLIN **auralises** programs by representing each of the Pascal constructs:

WHILE, **REPEAT**, **FOR...TO**, **FOR...DOWNTO**, **IF**, **IF...ELSE**, **CASE** & **CASE...ELSE** by a unique musical signature tune.

Pascal constructs

The constructs fall into one of two categories: iterations and selections. An individual construct shares some characteristics with others of its type, but also differs in various ways. For instance, although all selection constructs selectively execute portions of code, the individual constructs achieve this in different ways (IF...ELSE and CASE for instance). Figure 1 shows the hierarchy of these Pascal constructs and how these features can be mapped to the musical tunes.

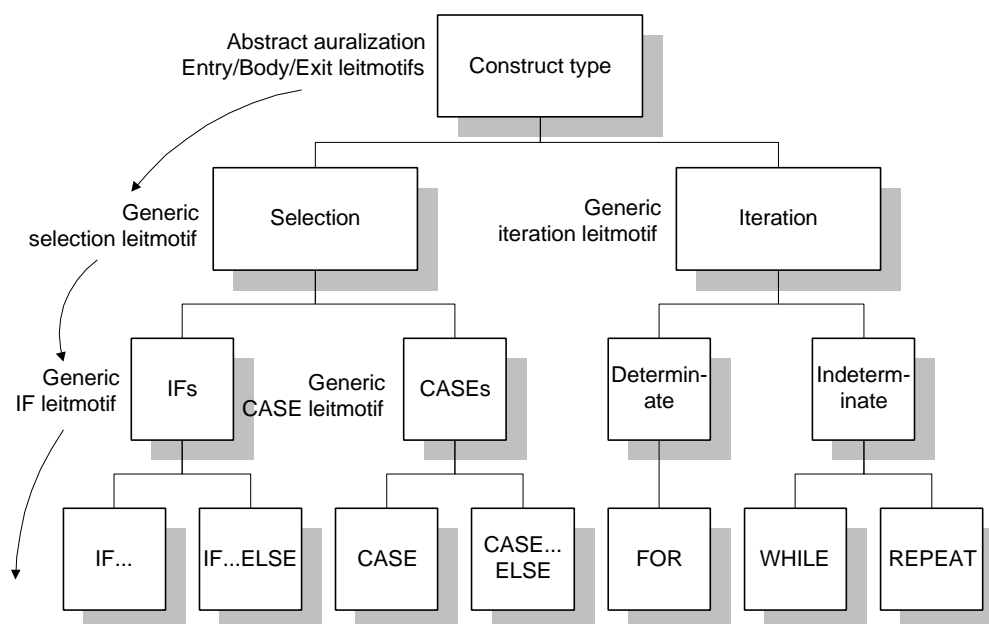


Figure 1 Hierarchy of Pascal Constructs

Design of tunes

For the two classes of construct (iteration and selection) there are four basic points of interest (POI) that require auralisation:

- entry to the construct (e.g., when the program enters a **WHILE** loop);
- evaluation of controlling condition (e.g., testing the condition belonging to the **WHILE**);
- execution of the construct's body (e.g., carrying the statements between the **WHILE**'s **BEGIN** & **END**);
- exit from the construct (e.g., the **WHILE** loop has finished).

Boolean Values

CAITLIN uses a major scale chord to represent the Boolean value True and a minor scale chord to represent the Boolean value False.

Construct Persistence & Drones

As all constructs may have large bodies (many, or time consuming statements inside them) then they may last for long periods of time. Therefore, to each construct has been added a background note (called a drone, as on the bagpipes) which plays continuously as long as the program is executing that construct. If several constructs are nested inside each other (e.g, a **FOR** loop within an **IF... THEN... ELSE**) then one drone for each construct will be heard.

Selections

A selection (e.g. **IF...THEN**) poses a question and then answers it. This is represented by a rising scale signifying entry to the selection construct (which when played sounds like a question being asked— just as we upwardly raise our voices when asking questions) and a descending scale denoting exit (showing that the question has been answered).

Iterations

We need to know when the loop returns to its starting point (that is, when it iterates) and when its controlling condition is tested. For the **REPEAT** and **WHILE** loops a simple major/minor chord is used when the loop condition is tested; this would be heard immediately after entry to the **WHILE** loop, but after the iterated statements of the **REPEAT** loop. Each time one of these chord devices is heard we know that the loop has reached its decision point. The null **WHILE** loop (where the controlling condition is *False* upon entry) would thus be heard as a sequence of: entry tune followed by minor chord for condition evaluation followed by the exit tune.

The **FOR** loops use a counter which takes incremental steps from a starting value to an end value. To denote this stepping up (or down) of the counter the pitch of the drone (background note) in the **FOR** loop is changed up to the next note of the scale (or down for the **FOR...DOWNTO**) for each repetition.

Variations on a theme

This idea of variations on a theme ensures that all selections sound like each other but can be distinguished by their individual mutations of the class motif.

Auditory parentheses

To help distinguish at the outset whether an auralisation represents an iteration or a selection we have used a device which we call auditory parentheses. Just as parentheses in writing serve to bracket in a sentence a related thought which is not part of the main text (for instance, here is an example), auditory parentheses provide sounds at the beginning and ending of constructs which serve to open and close the construct. For instance, we could transcribe a telephone conversation between Rosie and Jim by enclosing all of Rosie's words in round brackets and all of Jim's in square brackets. The different shaped brackets would identify who was speaking and where their speech began and ended. For example...

Rosie dials Jim's number. Telephone rings three times and is answered.
 [Hello?]
 (Hello Jim, it's me, Rosie.)
 [Oh, hi Rosie, it's nice to hear from you].
 3-second pause.
 (Sorry for the silence, I was yawning. Well, goodbye.)
 [Good bye.]
 Rosie replaces the handset.

In the same way, we have used a pair of sounds to mark the opening and closing of a selection construct and another pair of sounds to mark out iterations.

Other statements

Currently only the constructs have associated signature tunes. All other statements (e.g. assignments, procedure calls etc.) are represented simply by a single percussive

sound. There is one 'hit' per statement and all non-construct statements are sounded at the rate of one-per-beat. We will hear examples of this.

Examples

You will now hear several examples of auralisations.

1. Here are the auditory parentheses for *selections*. Start of a selection is marked by two cowbell sounds rising in pitch. The end of a selection is marked by the same two cowbell sounds, but this time the higher pitch comes first. This is intended to sound like a question and answer, as when we speak a question we tend to make our voice rise in pitch and when we speak an answer we lower the pitch.
2. Iterations are bracketed by a repeated sound. You will hear four hits on an open triangle to mark the beginning of an iteration. End of iterations is marked by another four hits but this time using a closed triangle sound.
3. A **WHILE** loop with 10 iterations.
4. A **WHILE** loop with zero iterations.
5. A **REPEAT** loop with 3 iterations.
6. An **IF** statement which yields a result of *True*.
7. An **IF...ELSE** statement which yields a result of *False*.
8. A **FOR...TO** loop with six iterations.
9. A **FOR...DOWNTO** loop with six iterations.
10. A **CASE** statement that yields no match.
11. A **CASE** statement that does give a match.
12. A **CASE...ELSE** statement that yields no match.
13. A **CASE...ELSE** statement that does yield a match.

Training Session

Feel free to make notes about each of the twenty auralisations you will hear.

- | | |
|-----------|-----------|
| 1. _____ | 11. _____ |
| 2. _____ | 12. _____ |
| 3. _____ | 13. _____ |
| 4. _____ | 14. _____ |
| 5. _____ | 15. _____ |
| 6. _____ | 16. _____ |
| 7. _____ | 17. _____ |
| 8. _____ | 18. _____ |
| 9. _____ | 19. _____ |
| 10. _____ | 20. _____ |

CAITLIN Experiment: Experience Questionnaire

Personal Details

1. Age _____
2. Gender _____
3. Cultural background (we are interested in this information since you may have been brought up in an environment where typical western music is not the norm.)

Education & Training

4. Highest educational qualification
 - A Levels
 - HNC/HND
 - Degree
 - Postgraduate Certificate/Diploma
 - Masters Degree
 - Doctoral Degree
 - Other
5. How many years of formal education/instruction have you had in computer programming?
 - Less than a year
 - 1 year
 - 2 or more years
6. In what programming languages have you written any code? (tick all that apply)
 - C
 - C++
 - Pascal
 - COBOL
 - FORTRAN
 - Ada
 - BASIC (including Visual BASIC)
 - Javascript / VB Script
 - Java
 - Any others

Musical Background

7. How would you classify your interest in music? (tick all that apply)

- I have no interest in music at all
 - I enjoy music as background as a listener or whilst dancing
 - I am very interested in music as a listener
 - I enjoy performing music on my own or with friends
 - I am a professional musician
8. Do you play a musical instrument?
- Yes
 - No

8a. If you answered yes to question 8, please specify:

Main instrument _____

Level of attainment

- Beginner
- Intermediate
- Semi-professional
- Professional

For how many years have you taken music lessons?

- I have not had formal lessons
- Less than 1 year
- 1 to 3 years
- More than 3 years

9. Do you sing?

- Yes
- No

If so, then at what level?

- I sing in the bath
- I sing informally to others
- I sing in a choir
- I sing semi-professionally
- I sing professionally

How many years of singing lessons have you had?

- None
- Less than a year
- 1 to 3 years
- More than 3 years

Musical Knowledge

10. What do the following musical terms mean?

Allegro _____

Presto _____

Rubato _____

Adagio _____

Crescendo _____

Fortissimo _____

11. In musical terms, what is a cadence?

12. Using the note Middle C as the root, or tonic note, what other notes are needed to play the following intervals?

Major 3rd _____

Perfect 4th _____

Perfect 5th _____

Seventh _____

Minor 3rd _____

Augmented 4th _____

13. How many quavers are there in a minim? _____

14. On a musical score, what does adding a dot after a note do to the note?

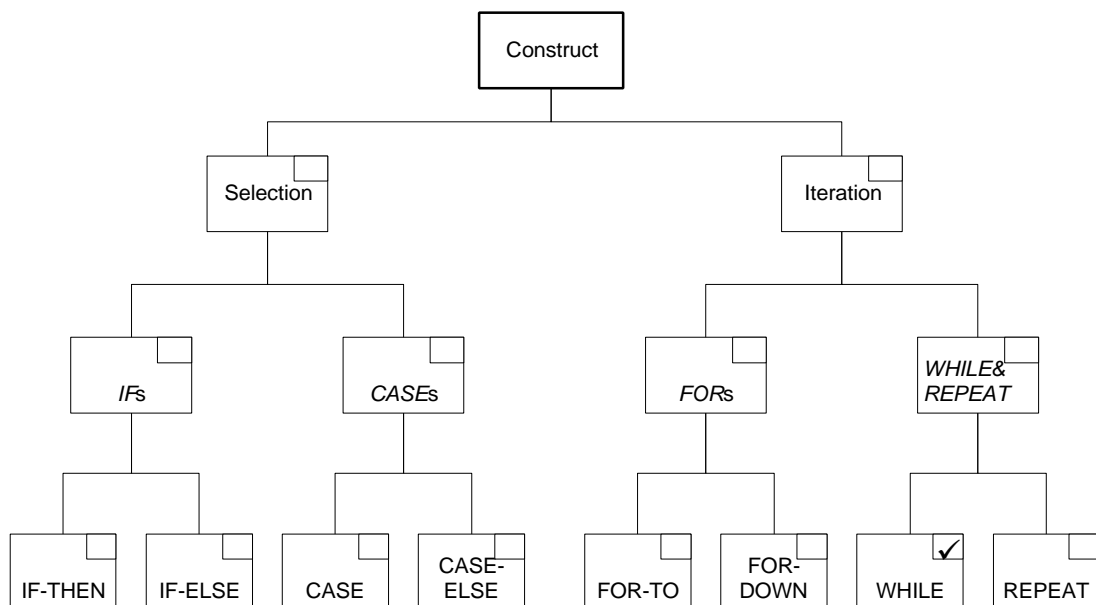
Test 1—Construct Identification

Introduction

In this test you will hear forty-two auralisations, each of which is of a single construct. An auralisation will be played three times, after which you should indicate which Pascal construct you think it represents by ticking the relevant box in the chart provided.

How to fill in the charts

For each auralisation you are required to identify it by ticking a box in the appropriate chart. There is one chart for each auralisation and the charts are numbered from 1 to 40. The charts look like this:



Notice that on the bottom level is a box for each of the constructs. On the next level up is a box giving the sub-class of the constructs (IFs, CASEs, FORs, WHILE/REPEAT). Above that is the name of the construct class (selection or iteration).

For a given auralisation, we want you to identify it as precisely as possible. If you feel you can identify a construct exactly, then tick one of the boxes on the **bottom** level. If you are unsure of the exact identity, but are confident that you know the *sub-class* then tick a box on the **third** level. If you cannot identify even the subclass but can at least identify the construct as either an iteration or a selection, then tick one of the two class boxes on the **second** level. If you have no idea at all, then leave the chart blank.

Examples

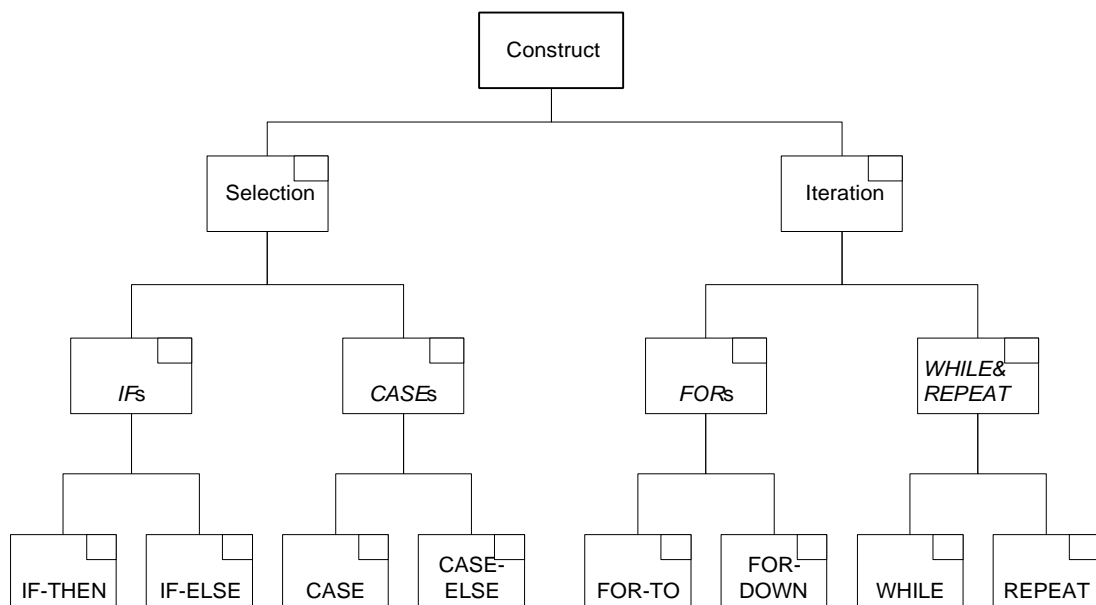
Here are four examples of how you might give your answers.

1. You listen to an auralisation and believe it to be a **CASE...ELSE** statement. Therefore, you would tick the box on the bottom level labelled **CASE- ELSE**.
2. You hear an auralisation which you believe to be one of the **FOR** loops, but you can't be sure whether it's a **FOR...TO** or a **FOR...DOWNTO**. Therefore, you would tick the third-level box labelled **FORs**.
3. This time you hear an auralisation which you are sure is a selection, but you cannot tell whether it is one of the **IF** statements or one of the **CASE** statements. This time you would tick the second-level box labelled **Selection**.
4. You listen to an auralisation and you cannot tell even whether it is an iteration or selection. This time you would not tick any box.

Exercises

You will now hear the forty auralisations in turn. Each one will be played three times and will be announced. Please indicate your answers on the numbered charts that follow.

Exercise 1



And so on...

Test 2—Nested & Sequential Constructs

Introduction

Obviously, program auralisations need to be applied to actual Pascal programs. The nature of most programs is that they contain a number of constructs, some of which will be sequential (one after the other), and some of which will be nested (one within the other)

Consider the following program

```

PROGRAM Table ;
USES
  crt ;
VAR
  i,
  j          : Integer ;
  numbers    : ARRAY [1 .. 4, 1 .. 3] OF Integer ;
  input_file : Text ;
  column_hi_total,
  column_low_total : Integer ;

BEGIN { Table }
1  ClrScr ;
2  Assign (input_file, 'table.dat') ;
3  Reset (input_file) ;
4  FOR i := 1 TO 4 DO
5    FOR j := 1 TO 3 DO
6      Read (input_file, numbers [i, j]) ;
7  Close (input_file) ;
8  FOR j := 1 TO 3 DO
9    BEGIN
10   column_hi_total := 0 ;
11   column_low_total := 0 ;
12   FOR i := 1 TO 4 DO
13     IF numbers [i, j] > 100 THEN
14       column_hi_total := column_hi_total + numbers [i, j]
15     ELSE
16       column_low_total := column_low_total + numbers [i, j] ;
17   Writeln ('High sales total for column ', j, ' is ', column_hi_total);
18   Writeln ('Low sales total for column ', j, ' is ', column_low_total);
19   END ;
END { Table }.

```

Notice that on line 4 there is a **FOR...TO** loop and nested within this is another **FOR...TO** loop. Following this nested section comes another **FOR...TO** loop. We would say then that at the top level of this program we have a sequence of two **FOR** loops. Also, notice in this second loop construct is nested another **FOR...TO** loop and within that is an **IF...THEN...ELSE**.

The auralisation system communicates whether or not constructs are nested. It does this by starting to play a continuous *drone* note at the start of the construct. This drone is not turned off until the construct ends. Thus, if another construct starts *within* the first, then we will hear it start *while* the drone is playing. This second construct will itself have a drone and so on.

The drones are organised so that the outer construct in a nested set has a lower pitch drone whilst those within will have higher pitches.

Constructs of equal depth (e.g., two sequential constructs nested within another, such as two IF statements within a single WHILE loop) would possess a drone of the same pitch.

Examples

You will now hear six example auralisations, three of which contain only sequential constructs, the remaining three containing only nested constructs.

1. A FOR...TO nested within a FOR...TO
2. A WHILE loop followed by a CASE statement.
3. An IF...THEN followed by an IF...THEN...ELSE
4. An IF...THEN nested within an IF...THEN...ELSE (i.e., an IF...THEN...ELSE...IF...THEN structure)
5. A CASE statement nested within a REPEAT loop
6. A FOR...DOWNTO loop followed by a FOR...TO loop.

Exercises

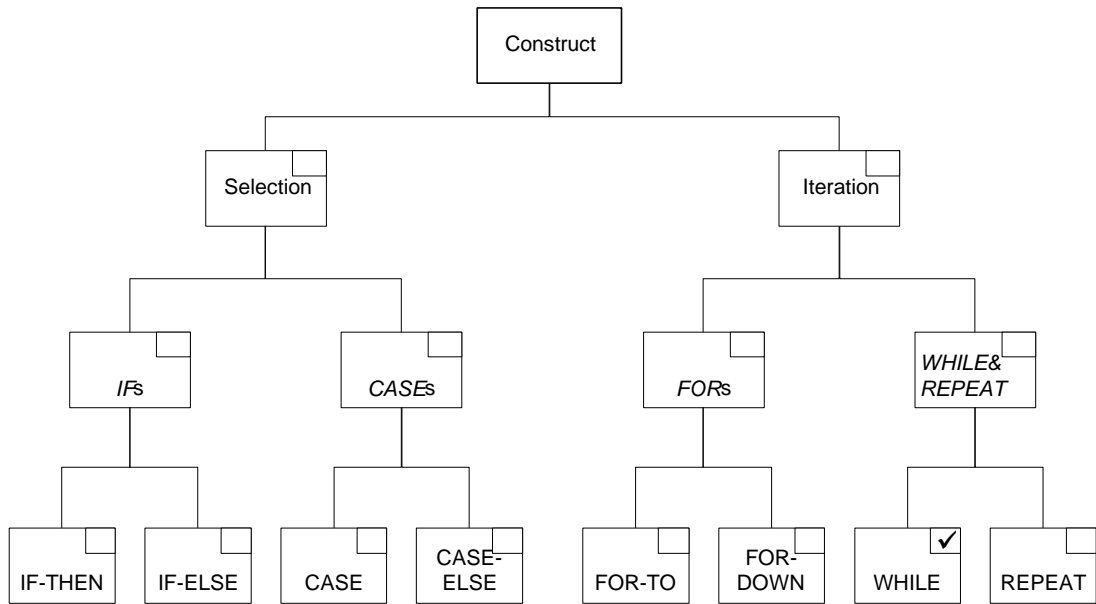
You will now hear ten auralisations. Each auralisation contains **two** constructs. These constructs will either be *sequential* (the second following the first) or *nested* (the second nested within the first).

As in the previous test you are required to identify the constructs you hear using the charts below. Additionally, you should state whether you think the auralisation is nested or sequential by ticking the appropriate box.

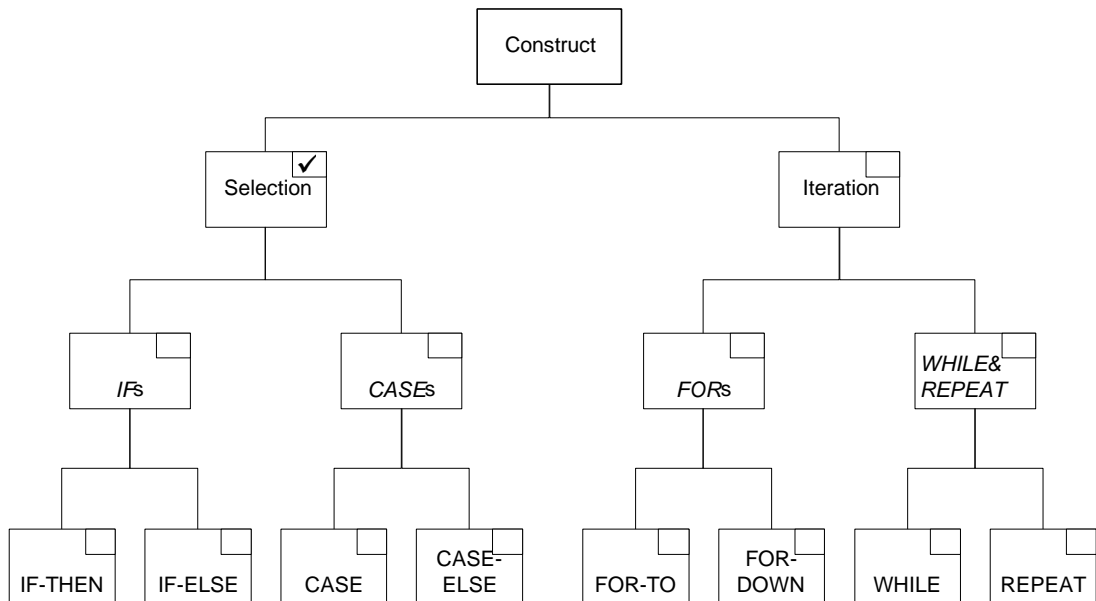
For example, if you heard an auralisation which you thought contained a selection of some description (you were unsure what type of selection it was) *nested within* a WHILE loop, then you would fill in the charts shown on the example over the page. Note, the **first** chart is for the first auralisation you hear and the **second** chart for the second auralisation regardless of whether they are nested or sequential.

Example Exercise

First Construct



Second Construct



This auralisation is:

- Nested
- Sequential