REVISE BTEC NATIONAL Computing

REVISION WORKBOOK

Pearson

REVISE BTEC NATIONAL

Computing

REVISION WORKBOOK

Series Consultant: Harry Smith

Authors: Steve Farrell, Mark Fishpool, Christine Gate and Richard McGill

A note from the publisher

While the publishers have made every attempt to ensure that advice on the qualification and its assessment is accurate, the official specification and associated assessment guidance materials are the only authoritative source of information and should always be referred to for definitive guidance.

This qualification is reviewed on a regular basis and may be updated in the future. Any such updates that affect the content of this Revision Workbook will be outlined at **www.pearsonfe.co.uk/BTECchanges.**

Pearson

Introduction

This Workbook has been designed to help you revise the skills you may need for the externally assessed units of your course. Remember that you won't necessarily be studying all the units included here – it will depend on the qualification you are taking.

BTEC National Qualification	Externally assessed units
Extended Certificate Foundation Diploma	1 Principles of Computer Science 2 Fundamentals of Computer Systems
Extended Diploma	1 Principles of Computer Science 2 Fundamentals of Computer Systems 3 Planning and Management of Computing Projects 4 Software Design and Development Project

Your Workbook

Each unit in this Workbook contains either one or two sets of revision questions or revision tasks, to help you **revise the skills** you may need in your assessment. The selected content, outcomes, questions and answers used in each unit are provided to help you to revise content and ways of applying your skills. Ask your tutor or check the Pearson website for the most up-to-date **Sample Assessment Material** and **Mark Schemes** to get an indication of the structure of your actual assessment and what this requires of you. The detail of the actual assessment may change so always make sure you are up to date.

Often, you will also find one or more useful features that explain or break down longer questions or tasks. Remember that these features won't appear in your actual assessment!

Grey boxes like this contain **hints and tips** about ways that you might complete a task, interpret a brief, understand a concept or structure your responses.

Guided

This icon will appear next to an example partial answer to a revision question or revision task. You should read the partial answer carefully, then complete it in your own words.

This is a **revision activity**. It will help you understand the processes or steps you could take in completing a revision question or task.

Links These boxes will tell you where you can find more help in Pearson's BTEC National Revision Guide. Visit www.pearsonschools.co.uk/revise for more information.

There is often space on the pages for you to write in. However, if you are carrying out research and making ongoing notes, you may want to use separate paper. Similarly, some units will be assessed through submission of digital files, or on screen, rather than on paper. Ask your tutor or check the Pearson website for the most up-to-date Sample Assessment Material to get an idea of the structure of the assessed exams or tasks and what is required of you.

Contents

Unit 1:
Principles of Computer Science

Your exam

Unit 1 will be assessed through an exam, which will be set by Pearson. You will need to use your computational thinking skills to solve computing problems through your response to questions that require short and long answers.

Your Revision Workbook

This workbook is designed to **revise skills** that might be needed in your exam. The selected content, outcomes, questions and answers are provided to help you to revise content and ways of applying your skills. Ask your tutor or check the **Pearson website** for the most up-to-date **Sample Assessment Material** and **Mark Scheme** to get an indication of the structure of your actual exam and what this requires of you. The details of the actual exam may change so always make sure you are up to date.

To support your revision, this workbook contains revision questions to help you revise the skills that might be needed in your exam.

Your response to the questions will help you to revise:
- computational thinking
- standard methods and techniques used to develop algorithms
- programming paradigms
- types of programming and markup languages.

Links To help you revise skills that might be needed in your Unit 1 exam, this workbook contains two sets of revision questions starting on pages 2 and 22. The first is guided and models good techniques, to help you develop your skills. The second gives you the opportunity to apply the skills you have developed. See the introduction on page iii for more information on features included to help you revise.

Revision test 1

To support your revision, the questions below help you to revise the skills that you might need in your exam. The revision test is divided into four questions, each based on a different scenario. You will need to refer to the information sheets on pages 14–21 in order to answer some of the questions. The details of the actual exam may change, so always make sure you are up-to-date. Ask your tutor or check the Pearson website for the most up-to-date Sample Assessment Material to get an idea of the structure of your exam and what this requires of you.

Links Please refer to Sections 1 and 2 of the information sheets on pages 14–16 in order to answer Revision Question 1.

Guided

1 Tanya is creating a 2D computer game. The user drives a taxi around the screen. Success is measured by how much money has been earned and how few penalty points have been added to the driving licence. Variables are used to hold both of these quantities.

A design for the Level 1 screen and the Level 1 design criteria are given in Section 1 of the information sheets on page 14.

Driving licence points will be given as variables.

(a) Identify **three** features of the game proposal, other than driving licence points, that would be represented as a variable. **3 marks**

1 Taxi sprite X coordinate

2 ....

3 ....

The variables will be anything inside the program where the value changes as the game runs.

During part of the play the user picks up passengers at £25, £20, £15, £12, £28 and £13, and has a driving fine of £50. The amount of money at the start of this play was £65.

(b) Calculate how much money was available at the end of this play.
You are advised to show your working. **2 marks**

Start of play money + (passenger fares) – driving fine =

£65 +

You will need to show your workings for the calculations you made in reaching your answer.

(c) Produce pseudocode that describes the movement of Taxi sprite when the user presses the right arrow key. **4 marks**

Your pseudocode needs to include keywords (in UPPER case) and to carefully sequence the actions. Use an indent for the actions following a **structure keyword** such as IF. (Don't forget to remove it for the next structure keyword).

Guided

```
BEGIN
IF key held down
  Increase speed
```

....

....

....

Links You can find more information on producing pseudocode in the Revision Guide, pages 9, 10 and 141.

Tanya writes some code to handle the keyboard inputs needed to move the Taxi sprite, which is shown in Section 2 of the information sheets on page 15.

(d) Identify **two** examples of duplicated code in this program section which responds to the Ctrl key. How can these duplicate codes be simplified? 4 marks

You will need to follow the code carefully to be able to answer this revision question. Look for duplicated code that can be reduced.

Example 1

Code responding to the Ctrl key is duplicated in lines 6–14, 18–26, 30–38, 42–50. This code is only needed once at the start of the subroutine.

Example 2

(e) Identify **two** examples of poor code in this section of programming, which mean the program is unlikely to work as intended. Explain how to improve the effectiveness of this code. 4 marks

Follow all the conditional statements (IF) to ensure that they will all work as intended.

1 The duplicated code responding to the Ctrl key in lines 6–14, 18–26, 30–38, 42–50, is flawed as it cannot reduce MoveDistance. The code needs to respond to

2

Programmers can use flow charts to plan the logic for their programs. Tanya has produced a flow chart (Figure 3 on page 16) to show the logic for:

- actions when the Ctrl key is pressed
- checking when a movement would collide the Taxi sprite with a wall
- responding to entry into a square (this should be shown as a process box in this flow chart, with no need to include the detailed logic).

(f) Identify an example of each of the following in the flow chart, by writing text from a flow chart symbol into your answer.

4 marks

You will need to identify the correct symbol, then carefully copy the words in the box into your answers.

Decision

Ctrl key pressed?

Input/output

....

Process

....

Start/end

....

(g) State the conventions for using the flow arrows in a flow chart.

2 marks

Flow lines connect the symbols together in a flow chart.

The default directions of flow are to the right or downwards.

Each flow line should have an arrowhead at one end which

....

....

....

Links There are two important aspects of producing a flow chart. It needs to use the standard BCS flow chart symbols and to accurately represent the flows (pathways) that are possible through a program. The flows should default as down or right, with arrows to confirm the direction to follow.
Be careful to show the decision questions and to label the routes out of them (for example YES and NO).
You can find more on how to produce a flow chart on pages 11 and 140 of the Revision Guide.

Total for Revision Question 1 = 23 marks

Links Please refer to Section 3 on page 17 in order to answer Revision Question 2.

Guided **2** Aarav runs dance classes. He is creating a program to administer the start every three months, with three sessions a week at different locations on different days. Standard membership entitles a customer to attend 12 sessions on days of their choice during the three months.

Part of the programming code is given on page 17.

The programming code Aarav created contains at least one bug, so he decides to use input boxes for rapid data entry to test the outcomes.

(a) Give the expected outputs from lblMemberFee. Find the text for these data entries (with the actual outcomes the code would produce) by completing the table. **3 marks**

First input	**Second input**	**Expected output**	**Actual output**
n	S	35	35
y	S	28	
N	M	50	50
Y	M	40	
N	p	75	
Y	p	60	

Program code is very precise and will only do the exact actions written. Be careful with individual characters: upper and lower case versions are treated as different.

You will need to fill in all the blank boxes in the 'Expected output' and 'Actual output' columns in the table.

(b) Name the control structures used in Aarav's code with their key words. **2 marks**

Loop (UNTIL)

....

The other structure will be a branch.

(c) Describe how a programmer can avoid input errors from capitalisation, for example, if the user types in 'n' rather than an expected 'N'. **3 marks**

Capitalisation becomes important for short inputs such as single characters or acronyms which are used in comparisons by code. If a text box is used for input ....

....

....

....

....

The programmer has a choice of whether to use a control that has a defined output, such as a checkbox, or to allow the user to type in a response, which will require code to allow for variations such as upper/lower case.

Attendance at sessions for each student are kept in a two-dimensional array called 'Bookings'.

(d) Explain why this is an appropriate data structure for this application. **4 marks**

The two dimensions of an array called 'Bookings' will provide an appropriate data structure for holding the attendances of members for the dancing sessions. One dimension, the rows, can be used for the members, and the other dimension, the columns, can be used for the sessions.

Your answer should be well structured and clear on the points you make. There are four marks for this revision question so you should include at least four points in your response.

....

....

....

....

....

....

Links The array is a data structure widely used in code as a variable containing several data items. For information on arrays, see the Revision Guide, page 23.

The program needs to sort the member attendance for a session into ascending order.

(e) Demonstrate how a bubble sort can be used to sort the data in the 'Before' row for the first dancing session by completing the table. **4 marks**

	Array(1)	Array(2)	Array(3)	Array(4)	Array(5)	Array(6)
Before	NJ1	GM2	RE1	JM2	JB3	GM1
	GM2	NJ1	RE1	JM2	JB3	GM1
	GM2	NJ1	JM2	RE1	JB3	GM1
					RE1	
						RE1
			NJ1			
	GM2	JM2	JB3	NJ1	GM1	RE1
	GM2	JB3	JM2	GM1	NJ1	RE1
	GM2	JB3	GM1	JM2	NJ1	RE1
After	GM1	GM2	JB3	JM2	NJ1	RE1

Complete the blank boxes in the Array(1)–Array(6) columns.

Links Revise how a bubble sort works on page 26 of the Revision Guide.

Aarav's programming code (see page 17) uses variables for testing the logic.

(f) Identify **two** variables in the code that could be declared as global variables and explain why this would be an appropriate scope structure for them. 3 marks

1 MemberFee could be suitable for declaring as a global variable. This would be an appropriate scope for this variable because the fee will be different for another member and the contents of this variable could be useful in other parts of the code.

You need to identify two variables which hold values that can be used in other parts of the program. Variable 1 above identifies MemberFee. There are another two potential global variables in the code given in the information pages on page 17.

You could start by identifying which of these variables would be better declared as constants.

2 ....

....

Links A global variable exists everywhere in the code and only ceases when the app closes. To revise global variables, see the Revision Guide, pages 14 and 154.

Aarav's programming code could use some global constants, rather than variables.

(g) State why a programmer would choose to use a **constant** in preference to a **variable**. 3 marks

A constant is very similar to a variable except that the data it holds does not normally change as the program runs.

Start by identifying the main difference between variables and constants.

....

....

....

....

Total for Revision Question 2 = 22 marks

Guided **3** A chef working in a gastropub writes a program to help her keep track of the recipes used in the kitchen. Each recipe has a type which may be a page-referenced in a cookery book, cut-out from a magazine, link to a web page, a hand-written note or printout.

When a new recipe is entered, the system allocates a reference based upon the type.

The chef has written some pseudocode to show the logic for how this reference is made (with comments at the end of some lines after single quotes).

```
BEGIN
INPUT Type                          'From a combo box

IF Type = "page reference"
   Ref = "PR"                       'page reference
ELSE
   Ref = "CO"                       'cut-out from a magazine
IF Type = "link to a web page"
   Ref = "WP"                       'link to a web page
IF Type = "hand-written notes"
   Ref ="HW"                        'hand-written notes
Ref = Random()                      'Add a random number to the reference
END
```

(a) Identify **two** errors in this pseudocode, and how each of the errors could be fixed. 4 marks

1 ELSE statement will not reliably respond to 'cut-out from a magazine' selected from the combo box.
Replace ELSE statement with IF Type = "cut-out"

The ELSE statement will wrongly identify all selections from the combo box, other than "page reference", as "cut-out from a magazine".
Identify another error and explain how to fix it.

2 ....

....

....

....

The program allocates a unique reference to each item, such as HW032 for a hand-written recipe, where the first two characters indicate the type.

(b) Explain **two** ways an item's reference can be used when the program runs. 4 marks

1 The first two letters of the reference can be used to sort items by type.

Answer (1) explains how the first part of the reference can be used by the program to sequence an array or other structure to separate out the types of reference.
How else could the first two letters of the reference be used by the program from user input?

2 ....

....

....

....

....

Guided

The chef decides to add a text box where the first two letters of a reference can be typed to search for that type of recipe.

(c) State **two** validation rules that could be applied to this text box. 2 marks

Rule

"CO" OR "HW" OR "PO" OR "PR" OR "WP"

Reason

You need to explain why this would be effective in preventing bad data from being entered.

....

(d) Discuss how single- or multi-dimensional arrays could be used to keep track of recipes for the chef. 6 marks

Single-dimensional arrays are a poor choice for this usage as at least three arrays would be needed to hold the reference number, description and where the recipe is located. Synchronising these into the same order would be a little more difficult than multi-dimensional arrays, especially if sorting needs to be coded into the program.

Multi-dimensional arrays are a good option, with a choice of ....

Your answer could start with a summary to confirm why you think arrays are an appropriate data structure for code to meet this problem.

....

....

....

....

....

....

....

....

Links A one-dimensional array has a single subscript inside the brackets and can be thought of as a single list of items. A multi-dimensional array has two (or more) subscripts inside the brackets and can be thought of as being able to hold a table of data. To revise arrays, see the Revision Guide, page 23.

The program used to keep track of recipes for the chef needs to be able to save the data to backup storage before it is closed and to retrieve this information when opened.

(e) Analyse how the program code will output the recipes' information to one or more data files and how code could input this data back into the program. **6 marks**

Program code needs to be able to output the recipes information to data file(s), otherwise any data in the arrays will be lost when the program closes. Similarly, this data needs to be input back into the program when it starts.

Continue the answer by analysing the types of loops that are most appropriate, using FOR to output the data, as the number of items are known, and any of the other loop types for input so they can continue iterating until the end of the data file.

Code to both output and input data will use loops to go through data ....

....

....

....

....

....

....

....

....

....

....

Total for Revision Question 3 = 22 marks

Links Please refer to Sections 4 and 5 of the information sheets on pages 18–21 in order to answer Revision Question 4.

Guided **4** Richard is creating a program to help him practise scales on a bass guitar.
A design for the screen and the design criteria are given in Section 4 of the information sheets on page 18. Code written to implement the program is shown in Section 5 on pages 19–21.

Program code uses structures to control program flow and to hold data.

(a) Name the variable types defined in these lines of code. **2 marks**

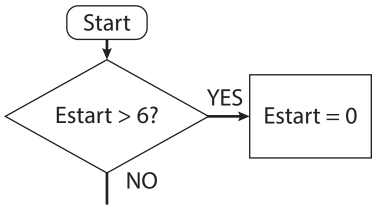
Line 4: Integer variable

Line 8: ……………………………………………………..

Make sure you look at the correct line numbers and include as much as you can about the variable defined on each of these lines of code.

(b) Draw a flow chart to represent lines 149 to 151 of Richard's code. **3 marks**

Be careful to represent which actions are dependent on the IF statement.



(c) Discuss any features of object orientated programming that could be used to code Richard's problem.

6 marks

Your answer should start with an overview of the features that are identified, followed by expansions of how each of these features could be utilised in the program code.

Object-orientated programming could be used to code Richard's problem using abstraction, inheritance and initial design for the program around real objects.

The program could be designed using an object-orientated approach, for example, the scales and fretboard being separate objects.

....

....

....

The first display could use abstraction to ....

....

....

....

Inheritance can be used to ....

....

....

....

(d) Evaluate the code Richard has written and produce suggestions on how this code can be improved.

12 marks

The code Richard has written can be improved in several ways.

Objects should be given meaningful names. The code has a button named Button1, which does not help this or future programmers ....

....

....

Look for other examples of poor coding and describe what you would do to rectify the problem.

The code should respond to ....

....

....

....

....

....

....

The code starting "G string" ....

Describe examples of any good practice you find.

Despite these issues, there are some examples of good practice.

Total for Revision Question 4 = 23 marks

END OF REVISION TEST 1

TOTAL FOR REVISION TEST 1 = 90 MARKS

Information for Revision test 1

The information below should be used to answer some of the revision questions on pages 2–13. The information is divided into five sections, with each section relating to a specific revision question. The details of the actual exam may change, so always make sure you are up to date. Ask your tutor or check the Pearson website for the most up-to-date Sample Assessment Material to get an idea of the structure of your exam and what this requires of you.

SECTION 1

Links The information in this section should be used to answer Revision Question 1(a) on page 2.

Figure 1 shows a design for the Level 1 screen from the game that Tanya has created.

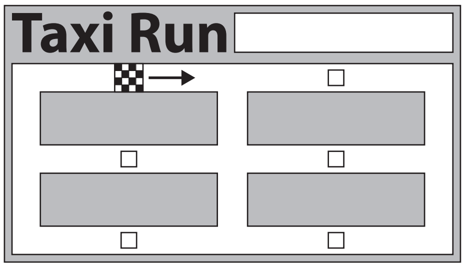


Figure 1

A sprite representing a taxi will start on the chequered grid.

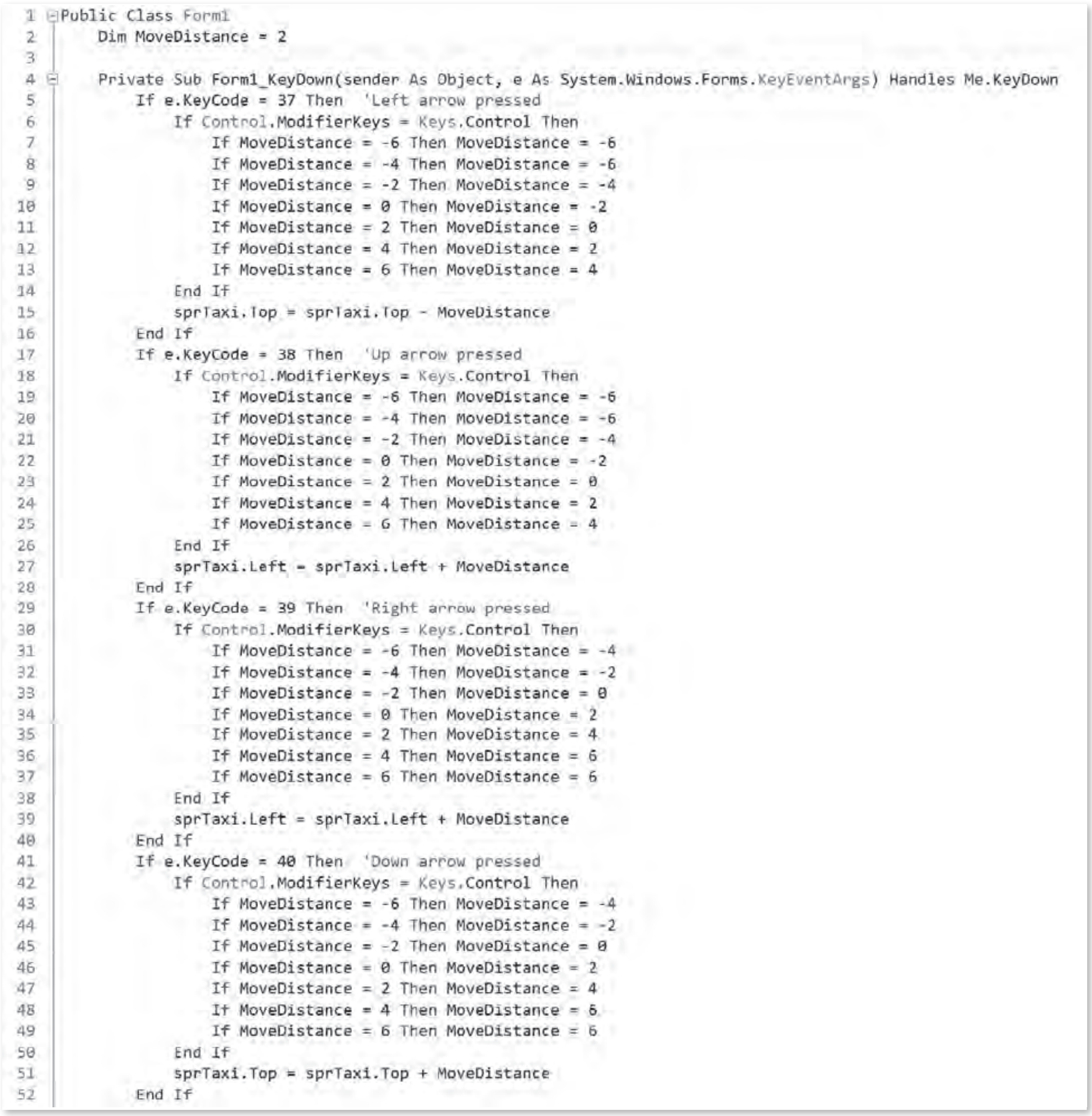
The screen for Level 1 of the game must meet the following design criteria.

- The taxi sprite will:
  - not be able to enter or cross any of the grey areas
  - keep moving in an XY direction set by the arrow keys
  - be accelerated or slowed by holding down an arrow key. The sprite moves 2, 4 or 6 units, matching the time the key has been held down.
- The white squares will:
  - change every second to become white, black or not visible
  - show white when they offer a positive outcome such as picking up a passenger
  - show black when they offer a negative outcome such as a speeding fine
  - have no impact when they are not visible.
- Progress will be tracked by:
  - money, which is:
    - increased by a new passenger
    - reduced by insurance, car tax, motoring fines
  - driving licence points, which are:
    - increased by motoring fines
    - reduced by a driving ban.

SECTION 2

Links The information in this section should be used to answer Revision Question 1(d) on page 3.

Figure 2 shows the code that Tanya first wrote to respond to key presses and to move the Taxi sprite.



```
Public Class Form1
    Dim MoveDistance = 2

    Private Sub Form1_KeyDown(sender As Object, e As System.Windows.Forms.KeyEventArgs) Handles Me.KeyDown
        If e.KeyCode = 37 Then  'Left arrow pressed
            If Control.ModifierKeys = Keys.Control Then
                If MoveDistance = -6 Then MoveDistance = -6
                If MoveDistance = -4 Then MoveDistance = -6
                If MoveDistance = -2 Then MoveDistance = -4
                If MoveDistance = 0 Then MoveDistance = -2
                If MoveDistance = 2 Then MoveDistance = 0
                If MoveDistance = 4 Then MoveDistance = 2
                If MoveDistance = 6 Then MoveDistance = 4
            End If
            sprTaxi.Top = sprTaxi.Top - MoveDistance
        End If
        If e.KeyCode = 38 Then  'Up arrow pressed
            If Control.ModifierKeys = Keys.Control Then
                If MoveDistance = -6 Then MoveDistance = -6
                If MoveDistance = -4 Then MoveDistance = -6
                If MoveDistance = -2 Then MoveDistance = -4
                If MoveDistance = 0 Then MoveDistance = -2
                If MoveDistance = 2 Then MoveDistance = 0
                If MoveDistance = 4 Then MoveDistance = 2
                If MoveDistance = 6 Then MoveDistance = 4
            End If
            sprTaxi.Left = sprTaxi.Left + MoveDistance
        End If
        If e.KeyCode = 39 Then  'Right arrow pressed
            If Control.ModifierKeys = Keys.Control Then
                If MoveDistance = -6 Then MoveDistance = -4
                If MoveDistance = -4 Then MoveDistance = -2
                If MoveDistance = -2 Then MoveDistance = 0
                If MoveDistance = 0 Then MoveDistance = 2
                If MoveDistance = 2 Then MoveDistance = 4
                If MoveDistance = 4 Then MoveDistance = 6
                If MoveDistance = 6 Then MoveDistance = 6
            End If
            sprTaxi.Left = sprTaxi.Left + MoveDistance
        End If
        If e.KeyCode = 40 Then  'Down arrow pressed
            If Control.ModifierKeys = Keys.Control Then
                If MoveDistance = -6 Then MoveDistance = -4
                If MoveDistance = -4 Then MoveDistance = -2
                If MoveDistance = -2 Then MoveDistance = 0
                If MoveDistance = 0 Then MoveDistance = 2
                If MoveDistance = 2 Then MoveDistance = 4
                If MoveDistance = 4 Then MoveDistance = 6
                If MoveDistance = 6 Then MoveDistance = 6
            End If
            sprTaxi.Top = sprTaxi.Top + MoveDistance
        End If
```

Figure 2

Figure 3 shows the flow chart that Tanya drew to help plan writing the code to move the Taxi sprite.

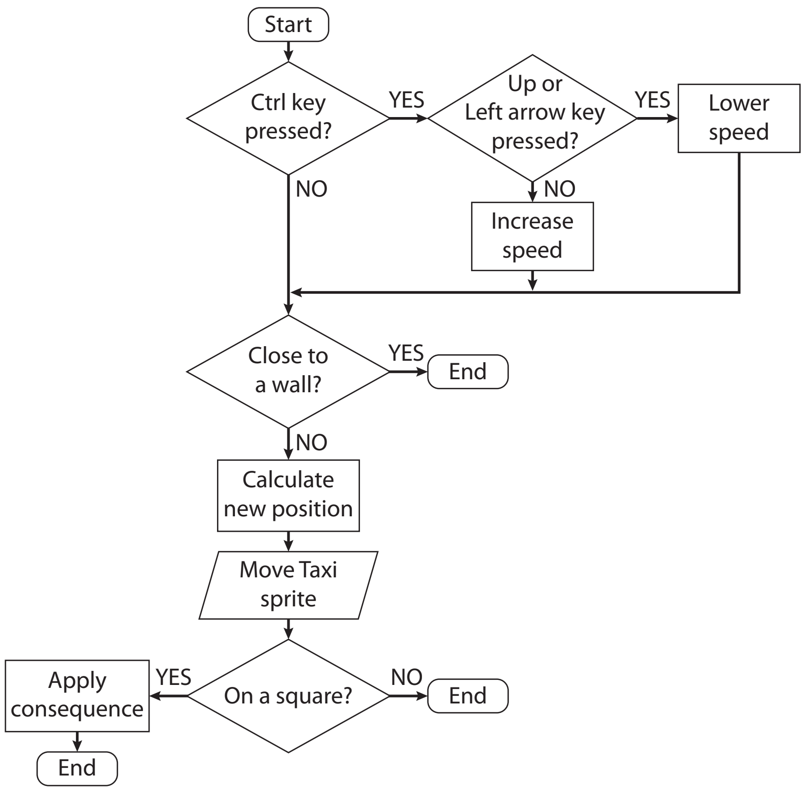


Figure 3

SECTION 3

Links The information in this section should be used to answer Revision Question 2(a) on page 5.

Figure 4 shows the code that Aarav wrote to help debug calculating membership fees.

```
Public Class Form1


    Private Sub btnTester_Click(sender As System.Object, e As System.EventArgs) Handles btnTester.Click

        Dim MemberFee, PreviousDiscount, RatePremium, RateProfessional, RateStandard As Single
        Dim MemberType, PreviousMember As Char
        Do Until (MemberType = "Q")
            MemberFee = 0
            PreviousDiscount = 0.8
            RateStandard = 35
            RatePremium = 50
            RateProfessional = 75
            PreviousMember = InputBox("Enter if a previous member")
            MemberType = InputBox("Enter member type" & vbCrLf & "(S=standard M=premium P=professional)")
            If PreviousMember = "Y" Then MemberFee = MemberFee * PreviousDiscount
            If MemberType = "S" Then MemberFee = RateStandard
            If MemberType = "M" Then MemberFee = RatePremium
            If MemberType = "P" Then MemberFee = RateProfessional
            lblMemberFee.Text = MemberFee
        Loop
    End Sub
End Class
```

Figure 4

SECTION 4

Links The information in this section should be used to answer Revision Question 4(a) on page 11.

Part of learning to play a musical instrument may involve practising scales – a set of musical notes played in sequence going up or down. Every scale has a root note where it starts, shown as black dots in Figure 5 below for a bass guitar. The scale notes are always the same distance apart.

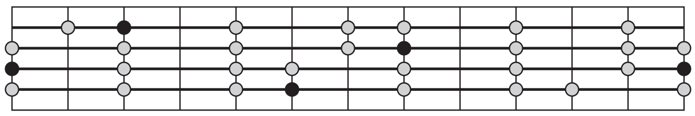


Figure 5

Richard is writing a Windows app to show where a scale can be played on the fretboard of a bass guitar. Figure 6 shows the screen design he has produced:

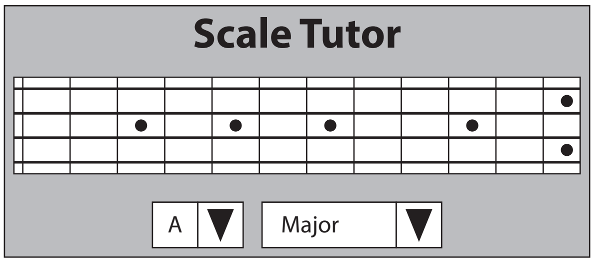


Figure 6

Combo boxes are to be used to select the key (e.g. A) and the scale (e.g. Major). Once selected, the positions on the bass fretboard where the scale is played are to be identified using labels which the code positions into the correct places. Figure 6 shows the positions for the scale of A Major.

SECTION 5

Links The information in this section should be used to answer Revision Question 4(b) on page 11.

Figure 7 shows the code that Richard first wrote to produce his program. It is written using VB.NET.

```
Public Class Form1
    Private Sub Button1_Click(sender As System.Object, e As System.EventArgs) Handles Button1.Click
        Dim GstringY = 65, DstringY = 110, AstringY = 155, EstringY = 200
        Dim I, Gstart, Dstart, Astart, Estart As Integer
        Dim CurrentG, CurrentD, CurrentA, CurrentE As Integer
        Static BeenThere = False

        Dim Scales(5, 6) As Integer
        Scales(0, 0) = 2
        Scales(0, 1) = 2
        Scales(0, 2) = 1
        Scales(0, 3) = 2
        Scales(0, 4) = 2
        Scales(0, 5) = 2
        Scales(0, 6) = 1

        Dim Xpos() As Integer = {20, 120, 220, 320, 420}
        Dim subsets() As Control = {lblN1, lblN2, lblN3, lblN4, lblN5, lblN6, lblN7, lblN8,
        Dim Amajor As New List(Of Point)

        'G string
        Gstart = 5
        CurrentG = 84
        Amajor.Add(New Point(CurrentG, GstringY))
        Gstart = Gstart + 1
        If Gstart > 6 Then Gstart = 0
        CurrentG = CurrentG + Scales(0, Gstart) * 84
        Amajor.Add(New Point(CurrentG, GstringY))
        Gstart = Gstart + 1
        If Gstart > 6 Then Gstart = 0
        CurrentG = CurrentG + Scales(0, Gstart) * 84
        Amajor.Add(New Point(CurrentG, GstringY))
        Gstart = Gstart + 1
        If Gstart > 6 Then Gstart = 0
        CurrentG = CurrentG + Scales(0, Gstart) * 84
        Amajor.Add(New Point(CurrentG, GstringY))
        Gstart = Gstart + 1
        If Gstart > 6 Then Gstart = 0
        CurrentG = CurrentG + Scales(0, Gstart) * 84
        Amajor.Add(New Point(CurrentG, GstringY))
        Gstart = Gstart + 1
        If Gstart > 6 Then Gstart = 0
        CurrentG = CurrentG + Scales(0, Gstart) * 84
        Amajor.Add(New Point(CurrentG, GstringY))
        Gstart = Gstart + 1
        If Gstart > 6 Then Gstart = 0
        CurrentG = CurrentG + Scales(0, Gstart) * 84
        Amajor.Add(New Point(CurrentG, GstringY))

        'D string
        Dstart = 2
        CurrentD = 0
        Amajor.Add(New Point(CurrentD, DstringY))
        Dstart = Dstart + 1
        If Dstart > 6 Then Dstart = 0
        CurrentD = CurrentD + Scales(0, Dstart) * 84
        Amajor.Add(New Point(CurrentD, DstringY))
        Dstart = Dstart + 1
        If Dstart > 6 Then Dstart = 0
        CurrentD = CurrentD + Scales(0, Dstart) * 84
        Amajor.Add(New Point(CurrentD, DstringY))
        Dstart = Dstart + 1
        If Dstart > 6 Then Dstart = 0
        CurrentD = CurrentD + Scales(0, Dstart) * 84
```

```
Amajor.Add(New Point(CurrentD, DstringY))
Dstart = Dstart + 1
If Dstart > 6 Then Dstart = 0
CurrentD = CurrentD + Scales(0, Dstart) * 84
Amajor.Add(New Point(CurrentD, DstringY))
Dstart = Dstart + 1
If Dstart > 6 Then Dstart = 0
CurrentD = CurrentD + Scales(0, Dstart) * 84
Amajor.Add(New Point(CurrentD, DstringY))
Dstart = Dstart + 1
If Dstart > 6 Then Dstart = 0
CurrentD = CurrentD + Scales(0, Dstart) * 84
Amajor.Add(New Point(CurrentD, DstringY))
Dstart = Dstart + 1
If Dstart > 6 Then Dstart = 0
CurrentD = CurrentD + Scales(0, Dstart) * 84
Amajor.Add(New Point(CurrentD, DstringY))

'A string
Astart = 6
CurrentA = 0
Amajor.Add(New Point(CurrentA, AstringY))
Astart = Astart + 1
If Astart > 6 Then Astart = 0
CurrentA = CurrentA + Scales(0, Astart) * 84
Amajor.Add(New Point(CurrentA, AstringY))
Astart = Astart + 1
If Astart > 6 Then Astart = 0
CurrentA = CurrentA + Scales(0, Astart) * 84
Amajor.Add(New Point(CurrentA, AstringY))
Astart = Astart + 1
If Astart > 6 Then Astart = 0
CurrentA = CurrentA + Scales(0, Astart) * 84
Amajor.Add(New Point(CurrentA, AstringY))
Astart = Astart + 1
If Astart > 6 Then Astart = 0
CurrentA = CurrentA + Scales(0, Astart) * 84
Amajor.Add(New Point(CurrentA, AstringY))
Astart = Astart + 1
If Astart > 6 Then Astart = 0
CurrentA = CurrentA + Scales(0, Astart) * 84
Amajor.Add(New Point(CurrentA, AstringY))
Astart = Astart + 1
If Astart > 6 Then Astart = 0
CurrentA = CurrentA + Scales(0, Astart) * 84
Amajor.Add(New Point(CurrentA, AstringY))
Astart = Astart + 1
If Astart > 6 Then Astart = 0
CurrentA = CurrentA + Scales(0, Astart) * 84
Amajor.Add(New Point(CurrentA, AstringY))

'E string
Estart = 3
CurrentE = 0
Amajor.Add(New Point(CurrentE, EstringY))
Estart = Estart + 1
If Estart > 6 Then Estart = 0
CurrentE = CurrentE + Scales(0, Estart) * 84
Amajor.Add(New Point(CurrentE, EstringY))
Estart = Estart + 1
If Estart > 6 Then Estart = 0
CurrentE = CurrentE + Scales(0, Estart) * 84
Amajor.Add(New Point(CurrentE, EstringY))
Estart = Estart + 1
If Estart > 6 Then Estart = 0
CurrentE = CurrentE + Scales(0, Estart) * 84
Amajor.Add(New Point(CurrentE, EstringY))
Estart = Estart + 1
If Estart > 6 Then Estart = 0
CurrentE = CurrentE + Scales(0, Estart) * 84
Amajor.Add(New Point(CurrentE, EstringY))
Estart = Estart + 1
If Estart > 6 Then Estart = 0
CurrentE = CurrentE + Scales(0, Estart) * 84
Amajor.Add(New Point(CurrentE, EstringY))
Estart = Estart + 1
```

```
        If Estart > 6 Then Estart = 0
        CurrentE = CurrentE + Scales(0, Estart) * 84
        Amajor.Add(New Point(CurrentE, EstringY))
        Estart = Estart + 1
        If Estart > 6 Then Estart = 0
        CurrentE = CurrentE + Scales(0, Estart) * 84
        Amajor.Add(New Point(CurrentE, EstringY))

        If Not BeenThere Then
            Amajor.Clear()
            BeenThere = True
            For Each s In subsets
                s.Visible = False
            Next
        Else
            BeenThere = False
        End If

        i = 0
        For Each p As Point In Amajor
            subsets(i).Location = p
            subsets(i).Visible = True
            i = i + 1
        Next

    End Sub

    Private Sub Button2_Click(sender As System.Object, e As System.EventArgs) Handles Button2.Click
        lblN10.Location = New System.Drawing.Point(400, 159)
    End Sub

    Private Sub Form1_Load(sender As System.Object, e As System.EventArgs) Handles MyBase.Load
        ComboBox1.Items.Add("A")
        ComboBox1.Items.Add("A#")
        ComboBox1.Items.Add("B")
        ComboBox1.Items.Add("C")
        ComboBox1.Items.Add("C#")
        ComboBox1.Items.Add("D")
        ComboBox1.Items.Add("D#")
        ComboBox1.Items.Add("E")
        ComboBox1.Items.Add("F")
        ComboBox1.Items.Add("F#")
        ComboBox1.Items.Add("G")
        ComboBox1.Items.Add("G#")

        ComboBox2.Items.Add("Dorian")
        ComboBox2.Items.Add("Harmonic minor")
        ComboBox2.Items.Add("Major")
        ComboBox2.Items.Add("Melodic minor")
        ComboBox2.Items.Add("Natural minor")
        ComboBox2.Items.Add("Pentatonic")

    End Sub
End Class
```

Figure 7