Q1
(a) Set aButton.Text To "Click Me"

(b) ReadOnly ensures that the user cannot change the value shown in the GUI, but they can still select parts of the text to copy. Code can still change the value shown in the text box.

(c)
stockCode As String
Set stockCode To codeTextBox.Text

(d) unit 1 P24 / 36 / HB P7
Set the TabIndex property of the control to 0 (if it is within any containers, such as group boxes or panels, ensure that their TabIndex is also set to 0) or set control's Focused property to True or add code to Form Load event to execute control's focus method.

Q2
(a) supply

(b) Yes. number, of type Integer

(c) number is not negative and is less than or equal to numberInStock

(d) Yes, the underlying field of the property NumberInStock will be reduced by number

(e)(i)
```
aShirt As Shirt
Set aShirt To New Shirt
Set aShirt.Code To "S12"
Set aShirt.LongSleeves To False
```

(ii)
```
If wanted > = 0 And wanted <= aShirt.NumberInStock Then
        aShirt.supply(wanted)
End If
```

Q3

Working 'Sketch' of the List to help follow the codes actions:

aList.add(-12) to aList.add(5)

0	1	2	3	4
-12	3	5	-3	5

aList.insert(1, 4)

0	1	2	3	4	5
-12	4	3	5	-3	5

aList.remove(5)

0	1	2	~~3~~	3	4
-12	4	3	~~5~~	-3	5

(Final values aList: -12, 4, 3, -3, 5 mystery = 3)

(a) 5

(b) False

(c) 3

(d) 12. The sum of the positive numbers in aList

Q4 Header is Method findFirst(word As String) As Integer

(a) Note that all the solutions below would return part words as well as whole words. It's not clear from the question whether this would be sufficient.

```
Return  Text.IndexOf(word)
```

Or

```
index As Integer
Set index To 0
found As Boolean
Set found To False
While index <= (Text.Length  - word.Length ) And Not found
        If word = Text.substring(index, word.Length) Then
                Set found To True
        Else
                Set index To index+1
        End If
End While
If Not found Then
        Set index  To -1
End If
Return index
```

Or (and this solution is left in just to show what a muddle you could get into trying to use indexOf in a solution)

```
Method  findFirst( word As String) As Integer
    index As Integer
    Set  index To 0
    returnIndex As Integer
    found As Boolean
    Set found To False
    endText As String
    Set endText To Text
    Set  index To endText.IndexOf(word.Chars(0))
    Set returnIndex To index
    While Not found And Not index = -1 And index <= endText.Length - word.Length
      If endText.Substring(index, word.Length) = word Then
        Set found To True
        Set returnIndex To returnIndex + index
      Else
        Set endText To endText.Substring(index + 1)
        Set index To endText.IndexOf(word.Chars(0))
      End If
    End While
    If found Then
      Return returnIndex
    Else
      Return -1
```

```
    End If
  End Function
```

Or (but also has multiple returns)

```
Method findFirst(word As String) As Integer
        position As Integer
        Set position To 0
        While ((position + word.length) <= Text.length)
                If (word = Text.substring(position, word.length)) Then
                        Return position
                End If
                position = position + 1
        End While
        Return -1
End Method
```

(b)

"sixpence"	word occurring in the middle of the sentence
"Sing" or "rye"	word occurring at either end
"barley"	word not occurring
"a"	word contained in Text returns the first occurrence of the word

Q5
(a)

The Vehicle constructor is protected which means it cannot be called by code in any class apart from Vehicle and its subclasses (derived classes). In addition Vehicle does not have a shared method which returns an instance of the class and no method of Car returns a new instance of Vehicle.

(b)

```
Method New As Car
        MyBase.New
        Set InsuranceClass To InsuranceType.Class1
End Method
```

(c)

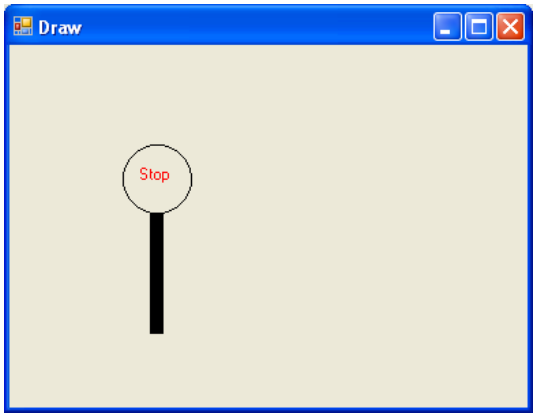
```
aCar As Car
Set aCar To New Car
Set aCar.Length To 2.2
Set aCar.InsuranceClass To InsuranceType.Class1F
```

(d) Overrides means that when the method ferryCharge is invoked on an instance of Car the method that will be executed will the one in Car, not the one with the same signature in the superclass. Overrides is used in a subclass to define different behaviour for that class for the method compared with the superclass. The ferryCharge method in the Vehicle class is not implemented, but in the Car class it is implemented.

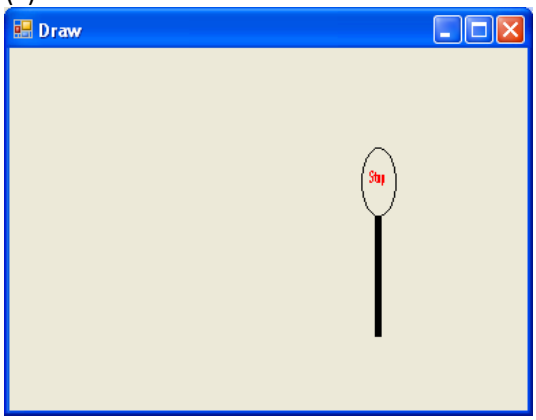
Q6
(a) HB p9 & 43

```
If saveDialog.ShowDialog = DialogResult.OK Then
        Try
                aRichTextBox.SaveFile(saveDialog.FileName)
        Catch ex As Exception
                'some error message
        End Catch
End If
```

(b)(i)



(ii)



The origin is moved to the point (200,0) and the x axis is scaled by 0.5 so that the drawing appears squashed horizontally by ½.

Q7
(a)

```
SELECT SeatingCode
 FROM Seating
WHERE Seats < 4;
```

(b) The SQL query returns a list of waiter codes and seats where the task duty is Clear. The list returned will be all the waiters that are clearing tables, and the number of seats at those tables. So here the resultant table will have 2 columns: Seats and WaiterCode and will consist of one row with values 6 and W respectively.

Seating INNER JOIN Task - Ref HB p55

SeatingCode	Seats	WaiterCode	Duty
T1	4	W1	Menu
T1	4	W2	Drinks
T2	6	W2	Clear

Q8

(a) & (b) Reference Unit 1 page 29 & HB P5

1	Effectiveness	There's no list of available goods so the user might not realise they could purchase some items	Provide ComboBox or ListBox of Items available
2	Efficiency	There's no way of selecting an item from those available to speed up completion of the form There's no way of selecting multiple items so the same item would have to be entered multiple times	ditto Have number of items NumericUpDownBox
3	Safety	There's no list of previously added items so the same item could be added twice by mistake or spelt wrongly The totalTextBox isn't read only, so the text can be changed	Provide ListBox as feedback for what has already been added to the list with number of items displayed. Modify the GUI to make the totalTextBox read only.
4	Utility		
5	Learnability		
6	Memorability	The user can't see what items they have already added.	Add a list box to show what items have been added so far

Q9
(a)

```
Method addNewPlayer(aName As String)
      aPlayer As Player
      Set aPlayer To New Player
      Set aPlayer.Name To aName
      Set aPlayer.Position To fPlayers.Count
      fPlayers.Add(aPlayer)
   End Method
```

(b)

```
ReadOnly Property NameList() As List(Of String)
      Get
         resultList As List(Of String)
         Set resultList To New List(Of String)
         For i As Integer From 0 To fPlayers.Count - 1
            resultList.Add(fPlayers.Item(i).Name)
         End For
         Return resultList
      End Get
   End Property
```

(c)(i)

```
 Private Method findPlayer(aNumber As Integer) As Player
      Return fPlayers.Item(aNumber)
 End Method
```

(ii) Effect of declaring findPlayer as Private is that the methods cannot be called outside the class. It is a helper method only.

(d)
(1) & (2)declare variables to hold winning and losing players
(3) & (4) declare variables as Integers to hold index of winning and losing players in fPlayers List
(5) & (6)Sets the player object of the winning and losing players based on the index of the winner and loser respectively in fPlayers
(7) If the winner is lower in the league than the loser
(8) & (9)record original Positions of winning player and losing player in winPos and losePos respectively
(10) change winning player's Position in league to losing player's Position: losePos
(11) loop through the list fPlayers representing all players in the league
(12) If next player's Position in league is higher than winner but lower than loser execute line 13
(13 move next player down one place in league by adding 1 to Position
(14) End If from line (12)
(15) End Loop from line 11
(16) move losing player one place down the league by adding 1 to Position
(17) End If from line (7)

Q10
VB Code shown below, followed by equivalent design code:

```
'part (a)(i)
    Private Sub updateImage()
        'Preconditions: None
        'Postconditions: FaceArea is updated to match the state of the face
sprite
      If fFace IsNot Nothing Then
        Dim g As Graphics
        g = Graphics.FromImage(FaceArea)    'HB p25
        g.Clear(Color.White)
        fFace.draw(g)
        g.dispose()
      End If
    End Sub
```

Method updateImage()
    If fFace IsNot Nothing Then
        g As Graphics
        Set g To Graphics.FromImage(FaceArea)
        g.Clear(Color.White)
        fFace.draw(g)
        g.dispose()
    End If
End Method

```
Public Sub checkAnswer(ByVal ans As Integer)
        'Preconditions: none
        'Postconditions: Moving is set to True. The face sprite is set to
smile or
        'frown according to whetehr answer is correct. The face sprite's
position is set to (0,0) and its direction of movement is set to 5 pixels
to the right,
        'so that the sprite is ready to move acrcoss the area given by
FaceArea.
        'The appropriate sound file is played. FaceArea is updated to mach
the
        'state of the face sprite
        fMoving = True
        fFace.Smile = IsCorrect(ans) 'True means smile, False means frown
        fFace.Position = New Point(0, 0)
        fFace.goRight(5)      'or setMovement(5, 0)

        'play sound
        If IsCorrect(ans) Then
            My.Computer.Audio.Play("applause.wav")
        Else
            My.Computer.Audio.Play("ooh.wav")
        End If
        updateImage()
    End Sub
```

Method checkAnswer(ans As Integer)
    Set fMoving To True
    Set fFace.Smile To IsCorrect(ans)

```
        Set fFace.Position To New Point(0, 0)
        fFace.goRight(5)
        If IsCorrect(ans) Then
                My.Computer.Audio.Play("applause.wav")
        Else
                My.Computer.Audio.Play("ooh.wav")
        End If
        updateImage()
End Method
```

```
  'part (b)(i)
    Public Sub updateView()
        facePictureBox.Image = fQuizAdmin.FaceArea
    End Sub
```

```
Method updateView()
        Set facePictureBox.Image To fQuizAdmin.FaceArea
End Method
```

```
'part (b) (ii)
    Private Sub OKButton_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles OKButton.Click
        fQuizAdmin.checkAnswer(Convert.ToInt32(ansNumUpDown.Value))
        faceTimer.Enabled = True
        updateView()
    End Sub
```

```
fQuizAdmin.checkAnswer(ansNumUpDown.Value)
Set faceTimer.Enabled To  True
updateView()
```

```
    'part (b) (iii)
    Private Sub newButton_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles newButton.Click
        fQuizAdmin.newExercise()
        updateView()
    End Sub
```

```
fQuizAdmin.newExercise()
updateView()
```

```
    'part (b) (iv)
    Private Sub faceTimer_Tick(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles faceTimer.Tick
        If fQuizAdmin.Moving Then
            fQuizAdmin.move()
        Else
            faceTimer.Enabled = False
        End If
        updateView()
    End Sub
```

```
If fQuizAdmin.Moving Then
```

```
        fQuizAdmin.move()
Else
        Set faceTimer.Enabled To False
End If
updateView()
```

Q11
Question 11
a)

	Name	Property	Initial Value
Controls			
ComboBox	depotCodeComboBox	DropDownStyle	DropDownList (N.B. Ensures ReadOnly)
		DataSource	depotBindingSource
		DisplayMember	DepotCode
TextBox	addressTextBox	ReadOnly	True
		DataBindings	Binding: Text, depotBindingSource, Address
TextBox	managerTextBox	DataBindings	Binding: Text, depotBindingSource, Manager
DataGridView	coachDataGridView	ReadOnly	True
		DataSource	FKCoachDepotBindingSource
		AllowUserToAddRows	False (see p33 of the handbook)
		AllowUserToDeleteRows	False (see p33 of the handbook)
TextBox	serviceTextBox	DataBindings	Binding: Text, fkCoachDepotBindingSource, LastService

b) i) The design code for the body of the OnClick event handler of the Save menu item is:

```
depotBindingSource.endEdit()
fkCoachDepotBindingSource.endEdit()
Try
        depotTableAdapter.update(coachDataSet.Depot)
        coachTableAdapter.update(coachDataSet.Coach)
        Messagebox.Show("Data Saved")
Catch ex As Exception
        MessageBox.Show("There has been an issue updating the database: "+ ex.Message)
End Try
```

ii) The design code for the body of the OnClick event handler of the Update button is:

```
depotBindingSource.endEdit()
fkCoachDepotBindingSource.endEdit()
```

iii) In the form closing event handler, call the save menu item event handler to ensure changes are saved to the database.

Alternative (longer solution – the previous version should be sufficient, but this version interacts with the user over the unsaved changes):

I would have a field indicating whether there is unsaved data, which is only set to true if an update is performed. If the user selects the save menu option then this flag is set back to false. When the application is closing, in the form's OnClosing event, I would check whether the field is true and if so prompt the user to save changes to the database – selecting yes would save the changes (by calling the save

menu item event handler), and no would discard them, and in either case the application will close afterwards.

c)

Requirement:	Comments:
The application should possess a way to view the depot address and manager name for each depot	These can be seen on the GUI in the address and manager fields, and are linked to the database so that they change as a different depot is selected.
The application should possess a way to edit the manager name at each depot	The manager text box is not read-only, and so can be edited. These updates can be saved back to the database with the save menu item.
The application should possess a way to view the data about coaches at a given depot	The data grid displays information about each coach in the depot.
The application should possess a way to edit the last service date for a given coach	The last service date text box is not read-only, so it can be changed, and subsequently saved back to the database using the save menu item.
The application should not allow any other data to be edited	The depot code combo is set to be a drop down list so new entries are not possible. The address text box is read-only so it cannot be modified by the user, and the data grid view is read-only and has properties set to prevent the user from adding new rows or deleting existing rows.