The Open University

MT264/C

Course Examination 2009
Designing applications with Visual Basic

Tuesday 16 June 2009 10.00 am – 1.00 pm

Time allowed: 3 hours

There are **TWO** parts to this examination. You should attempt **BOTH**.

You should attempt **ALL** the questions in Part 1.

In Part 2 there are three questions; marks will be given for your best **two** answers.

You are advised to spend about 100 minutes on Part 1 and 70 minutes on Part 2, and to leave yourself about 10 minutes for checking. Part 1 carries 64% of the total marks and Part 2 carries 36%.

You are advised to show all your working and to give reasons for all your answers unless a question is explicitly phrased 'Write down . . . '. You should begin each answer on a new page of the answer book.

At the end of the examination

Check that you have written your personal identifier and examination number on **each** answer book used. **Failure to do so will mean that your paper cannot be identified.**

Put all your used answer books together with your signed desk record on top. Fasten them in the top left corner with the round paper fastener. Attach this question paper to the back of the answer books with the flat paper clip.

The use of calculators is not permitted in this examination.

Part 1

*You should attempt **ALL** the questions in Part 1 and are advised to spend about **100 minutes** on it. This part carries 64% of the marks for the whole examination.*

There are eight questions, each worth 8 marks, in Part 1.

Question 1 (8 marks)

(a) Write down design code to make the Text property of a button aButton have the value "Click me".

(b) What is the effect of setting a TextBox control's ReadOnly property to True?

(c) Write down design code for the following tasks:

- declare a string variable stockCode;
- initialise stockCode to the Text property of a text box control codeTextBox.

(d) How can you ensure that a particular control has focus when an application is first opened?

Question 2 (8 marks)

Suppose that there is a class Shirt with the following description.

Class table: Shirt

Properties

Code **As** String

Description **As** String

LongSleeves **As** Boolean

NumberInStock **As** Integer

Constructor

New

Method

supply(number **As** Integer)

```
Constructor New
    'An object of type Shirt is created with Code and Description set to the
    'empty string, LongSleeves set to True and NumberInStock set to 0.
End Constructor

Method supply(number As Integer)
    'Preconditions: number is not negative and is less than or equal to
    'NumberInStock.
    'Postconditions: NumberInStock is reduced by number.
End Method
```

(a) What is the method identifier of the only method given above?

(b) Does the method have any parameters? If so, what type are they?

(c) Does the method have any preconditions? If so, what are they?

(d) Is it possible that the method changes the state of the object on which it is invoked?

(e) (i) Write design code to declare and create an object aShirt of type Shirt so that its Code property is "S12" and its LongSleeves property is False.

(ii) Suppose that wanted is an initialised Integer variable. Write design code to:

- check whether wanted satisfies the preconditions for the parameter of the supply method;
- if so, invoke the method supply on the object aShirt using the parameter wanted;
- otherwise, do nothing.

Question 3 (8 marks)

Suppose that the following design code has been executed.

```
aList As List(Of Integer)
mystery As Integer
Set aList To New List(Of Integer)
aList.add(−12)
aList.add(3)
aList.add(5)
aList.add(−3)
aList.add(5)
aList.insert(1, 4)
aList.remove(5)
Set mystery To 0
For Each number As Integer In aList
    If number > 0 Then
        Set mystery To mystery + number
    End If
End For
```

(a) Write down the value of aList.Count.

(b) Write down the value of aList.contains(−2).

(c) Write down the value of aList.Item(2).

(d) Write down the value of mystery, and briefly explain how you calculated it.

Question 4 (8 marks)

This question concerns the following class.

Class table: TextSearcher

Property
Text **As** String
Constructor
New
Method
findFirst(word **As** String) **As** Integer

```
Constructor New
    'A TextSearcher object is created, with Text set to the empty string.
End Constructor

Method findFirst(word As String) As Integer
    'Preconditions: word is not the empty string.
    'Postconditions: The position in Text of the first character in the first
    'occurrence of word in Text is returned. The position of the first character
    'in Text is 0. If word does not occur in Text, then −1 is returned.
End Method
```

(a) Write design code for the method findFirst. There is no need to include a comment.

You may wish to use the String method

substring(position **As** Integer, len **As** Integer)

This method returns the string that is formed from the string on which it has been invoked by taking the substring starting at index position and which has length len.

(b) Assume that Text is "Sing a song of sixpence a pocketful of rye". Give three examples of word with which to test your code. You should choose the examples such that each will test a significantly different aspect of your code. Briefly explain which aspect each example is intended to test.

Question 5 (8 marks)

This question concerns the following enumerated type InsuranceType, and two classes Vehicle and Car.

```
Enum InsuranceType
    Class1
    Class1F
    Class2
    Class3
End Enum
```

Class table: Vehicle

Properties

Length **As** Double

NumberOfWheels **As** Integer

Constructor

New Protected

Method

ferryCharge(passengers **As** Integer) **As** Double

```
Protected Constructor New
    'A Vehicle object is created, with Length and NumberOfWheels both
    'initialised to 0.
End Constructor

Method ferryCharge(passengers As Integer) As Double
    'Not implemented
End Method
```

Class table: Car Inherits Vehicle

Property

InsuranceClass **As** InsuranceType

Constructor

New

Method

ferryCharge(passengers **As** Integer) **As** Double Overrides

```
Constructor New
    'A Car object is created, with Length and NumberOfWheels both initialised
    'to 0 and InsuranceClass set to Class1 insurance type.
End Constructor

Overrides Method ferryCharge(passengers As Integer) As Double
    'Preconditions: passengers is not negative.
    'Postconditions: A value is returned giving the ferry charge
    '(given the number of passengers) for this instance of car.
End Method
```

(a) Briefly explain why client code cannot create an instance of Vehicle.

(b) Write down design code to implement the constructor of Car. There is no need to include a comment.

(c) Write a fragment of design code that:

- declares an instance aCar of type Car;
- creates the instance as a car of length 2.2 metres and insurance class Class1F.

(d) Explain what is meant by the qualifier **Overrides** for the ferryCharge method of Car.

Question 6 (8 marks)

(a) Write design code for the body of an event handler for a form. The code should enable a user to save the contents of a rich text box on the form, called aRichTextBox, to a file with name chosen by the user. You may assume that a SaveFileDialog control saveDialog has been added to the form. Part of your code should cause this dialogue box to be displayed.

There is no need to include a comment for the event handler.

(b) (i) Sketch or describe the effect of the following design code. You should assume that the code is in the OnPaint event handler of a panel, and that the panel is sufficiently large to contain the drawing. aFont is a standard font.

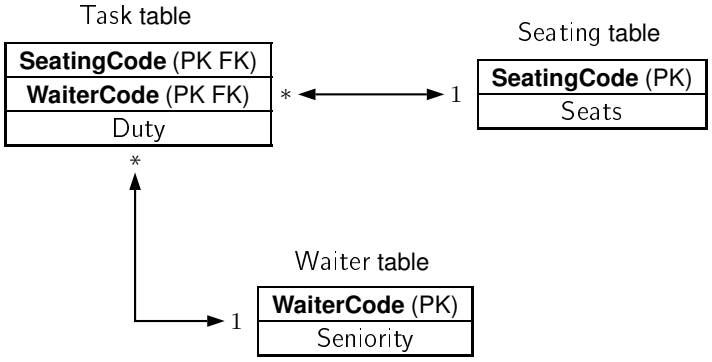
```
e.Graphics.drawEllipse(Pens.Black, 50, 50, 50, 50)
e.Graphics.fillRectangle(Brushes.Black, 70, 100, 10, 100)
e.Graphics.drawString("Stop", aFont, Brushes.Red, 60, 65)
```

(ii) How will the picture change if the following two lines of code are placed before the code given in part (i)?

```
e.Graphics.translateTransform(200, 0)
e.Graphics.scaleTransform(0.5, 1)
```

Question 7 (8 marks)

This question is about a database concerned with the allocation of waiters to tasks in a restaurant. Each task is linked to an area of seating, identified by a seating code. The database has the following ER diagram.



Some sample data is given in the tables below.

Task

SeatingCode	WaiterCode	Duty
T1	W1	Menu
T1	W2	Drinks
T2	W2	Clear

Waiter

WaiterCode	Seniority
W1	Waiter
W2	Leading waiter
W3	Head waiter

Seating

SeatingCode	Seats
T1	4
T2	6

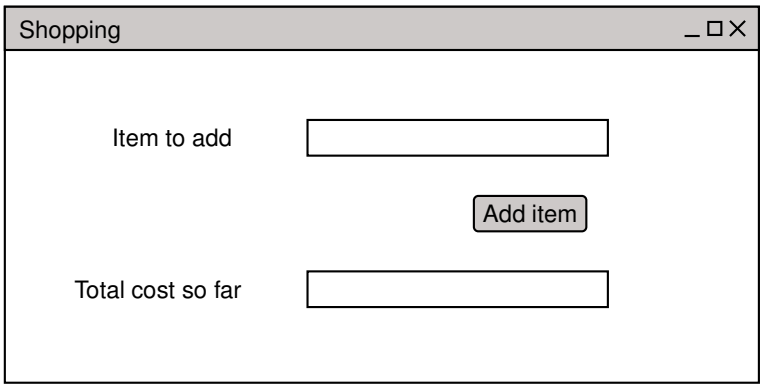
(a) Write an SQL statement that returns a table giving the seating codes for all the seating areas that seat fewer than 4 people.

(b) Describe in general terms the result table that will be returned by the following SQL statement.

```
SELECT Task.WaiterCode, Seating.Seats
FROM Seating INNER JOIN Task
ON Seating.SeatingCode = Task.SeatingCode
WHERE Task.Duty = 'Clear';
```

Question 8 (8 marks)

The following sketch and property table describe a graphical user interface for a simple shopping application. The user should enter a purchase by typing into the top text box and then click the *Add item* button to add this purchase to their shopping basket.



Property table: Shopping application

	Name	Property	Initial value
Container			
Form	MainForm	Text	Shopping
Controls			
Label	itemLabel	Text	Item to add
TextBox	itemTextBox	Text	
Button	addButton	Text	Add item
Label	totalLabel	Text	Total cost so far
TextBox	totalTextBox	Text	

This property table gives all of the design-time settings that have been made for the shopping application.

(a) Describe two problems with the usability of the application. You should name the usability goals involved.

(b) For each problem you have described, explain how the graphical user interface could be modified to improve the usability.

Part 2

*In Part 2, marks will be given for your best **two** answers, and you are advised to spend about **70 minutes** on it.*

Each question carries 18% of the marks for the whole examination, and an indication of the allocation of marks within each question is given in the margin.

Question 9 (18 marks)

This question is about a model for an application that maintains a sports ladder. Our sports ladder is based on a list of players who are taking part in games of a certain kind of sport, such as tennis or squash. Each time two players play each other, the result is recorded by updating player positions in the sports ladder as follows.

- If the player whose position was higher in the ladder wins, then no change is made.
- If the player whose position was lower in the ladder wins, then that player takes the position of the other player in the ladder. The losing player takes the position immediately below. To accommodate this change, the position of any player that prior to this game lay between the two players' positions is shifted down by one. For example, if there are players A, B, C, D and E in this order of positions, and D wins against B, then D takes B's position, B takes the position one below, and C is also shifted one down, resulting in the new ladder positions indicated by the order A, D, B, C and E.

The top player has position 0 in the ladder, the second highest player has position 1, and so on. New players are added at the bottom of the ladder.

The descriptions of the two classes to be used for the model are given below. Note that the order of the ladder is recorded purely by the Position property of the players. The Player instances are stored in a list fPlayers, in the order in which they initially get added to this list. For the purposes of this question, the index of a player in the list fPlayers will not change and will not usually coincide with the player's position in the sports ladder.

Class table: Player

Properties

Name **As** String

Position **As** Integer

Constructor

New

```
Constructor New
    'An instance of Player is created, with Name set to "" and Position set to −1.
End Constructor
```

Class table: LadderAdmin

Field

fPlayers **As** List(**Of** Player)

Property

NameList **As** List(**Of** String) ReadOnly; no field

Constructor

New

Methods

addNewPlayer(aName **As** String)

recordResult(indexOfWinner **As** Integer, indexOfLoser **As** Integer)

findPlayer(aNumber **As** Integer) **As** Player

'Other features not used in this question have been omitted.

```
Constructor New
    'An instance of LadderAdmin is created, with an empty list of players.
End Constructor

ReadOnly Property NameList As List(Of String)
    Get
        'Gives a list of names of the players in the order in which the
        'players were initially added to (and appear in) the list of players.
    End Get
End Property

Method addNewPlayer(aName As String)
    'Preconditions: aName does not yet occur in NameList.
    'Postconditions: A player with the given name is added to the end of
    'the list of players (so aName will appear at the end of NameList).
    'The player's position is set to the bottom of the sports ladder.
End Method

Method recordResult(indexOfWinner As Integer, indexOfLoser As Integer)
    'Preconditions: indexOfWinner and indexOfLoser are valid indexes of NameList.
    'Postconditions: The positions of the players in the ladder are updated
    'according to the result indicated by indexOfWinner and indexOfLoser.
    'Positions in the ladder of other players are changed as necessary.
End Method

Private Method findPlayer(aNumber As Integer) As Player
    'Preconditions: aNumber is a valid index of NameList.
    'Postconditions: The Player instance is returned that corresponds to the
    'name at index aNumber in NameList.
End Method
```

(a) Write design code for the method addNewPlayer. There is no need to include a comment.

Hint: As part of your answer, you should create a new instance of Player and add this player to the end of the list of players fPlayers. [*4*]

(b) Write design code for the property NameList. The index order of the names should be the same as that used for the corresponding players in the list of players fPlayers. There is no need to include a comment. [*5*]

(c) (i) Write design code for the private method findPlayer. There is no need to include a comment.

(ii) What is the effect of declaring findPlayer as **Private**? [*3*]

(d) The following design code implements the method recordResult of LadderAdmin. Briefly explain the effect of each line of code. The lines are numbered for ease of reference. [*6*]

```
(1)   winPlayer As Player
(2)   losePlayer As Player
(3)   winPos As Integer
(4)   losePos As Integer
(5)   Set winPlayer To findPlayer(indexOfWinner)
(6)   Set losePlayer To findPlayer(indexOfLoser)
(7)   If winPlayer.Position > losePlayer.Position Then
(8)       Set winPos To winPlayer.Position
(9)       Set losePos To losePlayer.Position
(10)      Set winPlayer.Position To losePos
(11)      For Each person As Player In fPlayers
(12)          If person.Position < winPos And person.Position > losePos Then
(13)              Set person.Position To person.Position + 1
(14)          End If
(15)      End For
(16)      Set losePlayer.Position To losePos + 1
(17)  End If
```

Question 10 (18 marks)

This question is about a simple arithmetic quiz application for a small child. The project specification is as follows.

> **Project specification: Simple quiz**
>
> There should be a way to display an arithmetical exercise and ways for the player to enter and confirm an answer (which will be an integer). There should also be a way of starting a new exercise.
>
> When the player confirms that they have entered an answer, there should be a response as follows. If the answer is correct, a sound file *applause.wav* should be played and a smiley face moved across the screen. If the answer is incorrect, a sound file *ooh.wav* should be played and a frowning face moved across the screen. In either case, the face should move horizontally from left to right and come to a halt just below the answer.

The smiley and frowning faces will be implemented using a sprite of type Face, which inherits from HVSprite. The class description is given below. The bitmap given by the Picture property is used in the draw method for drawing the smiley or frowning face. You will not be asked to implement any part of this class.

Class table: Face Inherits HVSprite

Properties	
Picture **As** Bitmap	ReadOnly
Smile **As** Boolean	WriteOnly
Constructor	
New	
Methods	
draw(g **As** Graphics)	Overrides
dispose()	

```
Constructor New
    'An instance of Face is created, with Picture set to a bitmap of a
    'smiley face and Smile set to True.
End Constructor

WriteOnly Property Smile As Boolean
   Set(value As Boolean)
       'Smile is set to value. Picture is updated appropriately.
   End Set
End Property

Overrides Method draw(g As Graphics)
    'Preconditions: none
    'Postconditions: Picture is drawn at Position using g.
End Method

Method dispose()
    'Preconditions: none
    'Postconditions: The bitmap is disposed of.
End Method
```

The way in which the arithmetical exercises will be generated has not been fully described and is of no concern to this question.

There will be a class QuizAdmin with the following description. The FaceArea bitmap is intended to be of the same size as the area through which the smiley or frowning face should move. The FaceArea property should be used to display the correct feedback in this area. It will give a bitmap that is either entirely white for no feedback, or that shows a smiley or frowning face on a white background in an appropriate position.

Class table: QuizAdmin

Field	
fFace **As** Face	
Properties	
Exercise **As** String	ReadOnly
FaceArea **As** Bitmap	ReadOnly
Moving **As** Boolean	ReadOnly
Constructor	
New(area **As** Size)	
Methods	
newExercise()	
checkAnswer(ans **As** Integer)	
move()	
updateImage()	Private
isCorrect(ans **As** Integer) **As** Boolean	Private
dispose()	
'Other features not used in this question have been omitted.	

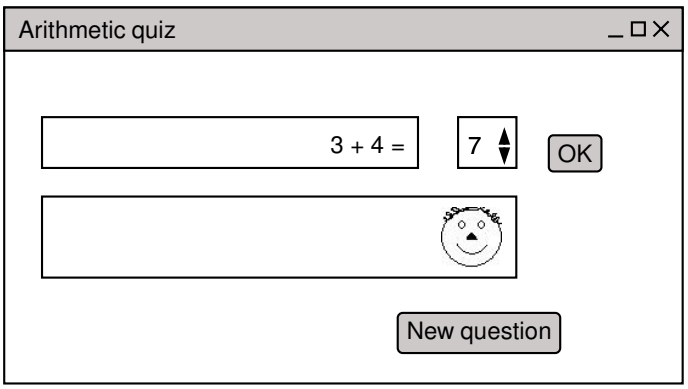
```
Constructor New(area As Size)
    'An instance of QuizAdmin is created, with the face sprite set to smile and
    'Exercise set to a new exercise. FaceArea is set to a white bitmap of size
    'area. Moving is set to False.
End Constructor

Method newExercise()
    'Preconditions: none
    'Postconditions: The Exercise property is updated to give a new exercise.
    'This question is not concerned with the details of how this is done.
    'Moving is set to False. FaceArea is updated to give no face, that is,
    'it is coloured entirely white.
End Method

Method checkAnswer(ans As Integer)
    'Preconditions: none
    'Postconditions: Moving is set to True. The face sprite is set to smile or
    'frown according to whether ans 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 across the area given by FaceArea.
    'The appropriate sound file is played. FaceArea is updated to match the
    'state of the face sprite.
End Method
```

```
Method move()
    'Preconditions: none
    'Postconditions: If Moving is True, the face sprite is moved. A check is made
    'on its position and Moving is set to False if appropriate. Otherwise nothing
    'is done. FaceArea is updated to match the state of the face sprite.
End Method

Private Method updateImage()
    'Preconditions: none
    'Postconditions: FaceArea is updated to match the state of the face sprite.
End Method

Private Method isCorrect(ans As Integer) As Boolean
    'Preconditions: none
    'Postconditions: True is returned if ans is the correct answer to Exercise.
    'Otherwise False is returned.
End Method

Method dispose()
    'Preconditions: none
    'Postconditions: The bitmap and face sprite are disposed of.
End Method
```

(a) (i) Write design code for the method updateImage of QuizAdmin. This will involve drawing the face sprite at its current position on the graphics object of the FaceArea bitmap. The rest of the bitmap should be coloured white. There is no need to include a comment.

(ii) Write design code for the method checkAnswer of QuizAdmin. You should make use of the private method isCorrect in your answer. There is no need to include a comment. [*10*]

(b) A possible graphical user interface and property table for the application are given below. The box just above the *New question* button is a picture box for displaying the smiley or frowning face.



Property table: Arithmetic quiz

	Name	Property	Initial value
Container			
Form	MainForm	Text	Arithmetic quiz
Controls			
TextBox	questionTextBox	ReadOnly	True
NumericUpDown	ansNumUpDown		
Button	okButton	Text	OK
PictureBox	facePictureBox		
Button	newButton	Text	New question
Component			
Timer	faceTimer	Interval	100
		Enabled	False
Field			
QuizAdmin	fQuizAdmin		

It has been decided that the timer should be enabled only while the smiley or frowning face is moving.

(i) Write design code for a form method updateView that ensures that the graphical user interface represents the current state of the model.

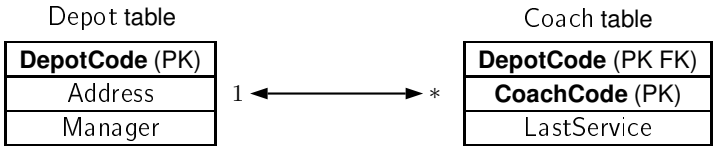
(ii) Write the design code body for the OnClick event handler of the *OK* button. There is no need to include a comment.

(iii) Write the design code body for the OnClick event handler of the *New question* button. There is no need to include a comment.

(iv) Write the design code body for the OnTick event handler of faceTimer. There is no need to include a comment. [*8*]

Question 11 (18 marks)

This question is about an application that gives access to a database concerning coaches and the depots to which they belong. The data records some details of the depots and the coaches. The ER diagram is shown below.



Some sample data is as follows.

Depot

DepotCode	Address	Manager
D1	Victoria, London	C.D. Jones
D2	Headrow, Leeds	A. Braithwaite

Coach

DepotCode	CoachCode	LastService
D1	LC1	20 September 2008
D1	LC2	8 May 2009
D2	YC1	4 April 2009

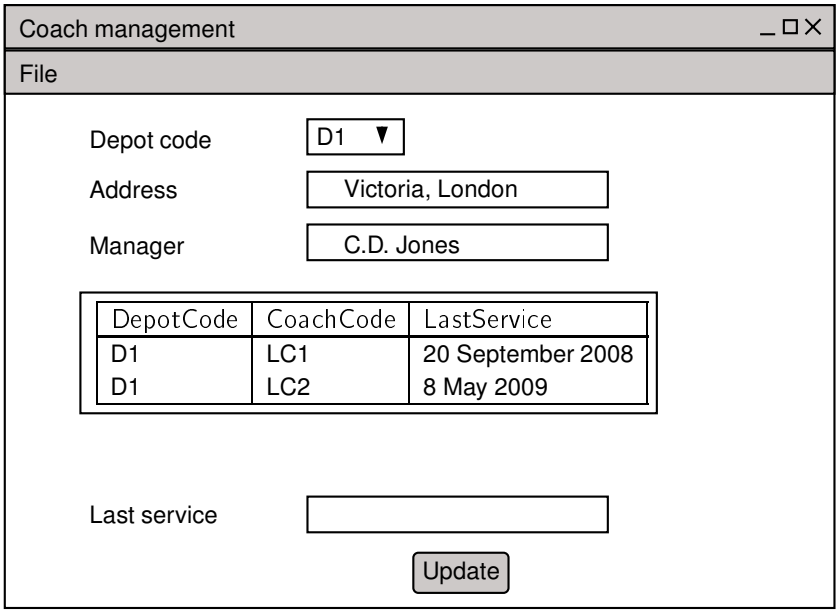
The specification and a possible GUI for the application are shown below.

Project specification: Coach management

The application is intended to allow the user to view and make some edits to the coaches database. The application should possess:

- a way to view the depot address and manager name for each depot;
- a way to edit the manager name at each depot;
- a way to view the data about coaches at a given depot;
- a way to edit the last service date for a given coach.

The application should not allow any data to be edited, other than that specified above.



Property table: Coach management

	Name	Property	Initial value
Containers			
Form	MainForm	Text	Coach management
MenuStrip	mainMenu		
Controls			
Label	depotCodeLabel	Text	Depot code
ComboBox	depotCodeComboBox		
Label	addressLabel	Text	Address
TextBox	addressTextBox		
Label	managerLabel	Text	Manager
TextBox	managerTextBox		
DataGridView	coachDataGridView		
Label	serviceLabel	Text	Last service
TextBox	serviceTextBox		
Button	updateButton	Text	Update
Components			
DataSet	coachDataDataSet		
BindingSource	depotBindingSource	DataSource	coachDataDataSet
		DataMember	Depot
DepotTableAdapter	depotTableAdapter		
BindingSource	fkCoachDepotBindingSource	DataSource	depotBindingSource
		DataMember	FKCoachDepot
CoachTableAdapter	coachTableAdapter		

Property table: Coach management – main menu

	Name	Property	Initial value
Container			
MenuStrip	mainMenu		
Controls			
Menu	fileMenu	Text	File
MenuItem	saveMenuItem	Text	Save
MenuItem	exitMenuItem	Text	Exit

(a) Write down an extension to the Controls section of the main property table shown above that gives details of the data bindings that can be set up during design time. Also include any other non-default properties that should be set to meet the given specification. You should in particular make sure that a user's editing permissions are set correctly. You do not need to include controls, such as labels, whose full details are given. [*6*]

(b) (i) Write the design code body for the OnClick event handler of the *Save* menu item. There is no need to include a comment.

(ii) Write the design code body for the OnClick event handler of the *Update* button. There is no need to include a comment.

(iii) How would you ensure that an attempt is made to save updated data to the underlying database when the application is closed? [*7*]

(c) Briefly describe how your application design fits the given project specification. In your answer you should address in turn each part of the specification, which is given in a box towards the beginning of this question. [*5*]

[END OF QUESTION PAPER]