Visual Basic Level 2

Information Technology
7261-225

PO A25036

7261/225
Visual Basic Level II

CONTENTS **PAGE**

INTRODUCTION TO THIS MODULE

This module is one of a series within the 7261 scheme in the broad area of computer programming and software and is at Level II.

This module aims to enable candidates to:

create user interfaces based on good design principles
utilise controls, properties, events and methods
use Visual Basic facilities to assist with the efficient production of code
create programs that validate and respond to user input
make use of constants and a range of variable types
develop an understanding of event-driven programs
use simple error handling techniques
create simple multi-form applications, including dialog boxes.

CANDIDATE ELIGIBILITY

The selection of candidates is at the discretion of centres. Candidates require some previous knowledge of Visual Basic to undertake this module. It would be advantageous for candidates to have completed 7261/205 Coding and Programming using Visual Basic Level I, prior to undertaking this module.

RESOURCE REQUIREMENTS

In order to undertake this module a candidate will require access to a microcomputer or work station running Windows Version 3.11 or later and Microsoft Visual Basic version 3.0 or later (Standard or Professional Edition).

ASSESSMENT PROGRAMME

To ensure that the acquisition and assessment of competence is enabled in a variety of ways, a number of alternative methods of assessment are acceptable.

The assessment programme requires work based assessment of practical competence together with confirmation that the candidate has acquired the necessary underpinning knowledge.

Because the creation of suitable assessment materials can be very time consuming for supervisors, trainers or teachers, each module in the 7261 series provides a comprehensive and standardised set of materials suitable for assessing both practical competence and underpinning knowledge. These are the preferred methods of assessment and they may be used either as they stand or as exemplars which define the standard of competence a candidate must achieve.

Standard set of assessment for this module:

The following must be completed satisfactorily by any candidate undertaking the standard assessment for this module:

Two practical assignments as follows:

Mandatory		PA 7261-225-01
1 **selected at**	(	PA 7261-225-02
random from	(	PA 7261-225-03
	(	PA 7261-225-04

One written MCQ test

Test 7261-225-A_

Alternative assessment

Equivalent, proven competence or certificated confirmation of equivalent competence may be accepted for exemption from the standard assessment material, but at least **one** of the practical assignments provided in the standard set must be undertaken. Detailed evidence of the equivalent competence achieved must be supplied when a certificate is requested.

For further details please refer to Scheme Notes Issue 6 (or later issue) or "Alternative Assessments for Information Technology Schemes" available from Division 13 at City and Guilds.

7261/225 VISUAL BASIC LEVEL II
CANDIDATE ASSESSMENT RECORD

CENTRE NAME _____ CENTRE NUMBER _____

CANDIDATE'S NAME _____ CANDIDATE'S NUMBER _____

TITLE (Mr/Ms) _____ CANDIDATE'S DATE OF BIRTH _____

DATE MODULE STARTED: _____

DATE MODULE COMPLETED: _____

DATE CERTIFICATE REQUESTED: _____

ASSESSMENT REFERENCE	ALTERNATIVE USED (tick if applicable)	DATE COMPLETED	TUTOR SIGNATURE	TUTOR NAME
7261-225-A_	Complete KA-1			
PA 7261-225-01	Complete P-1			
PA 7261-225-0_	Complete P-1			

TUTOR CONFIRMATION	DATE COMPLETED	TUTOR SIGNATURE	TUTOR NAME
LOG BOOK COMPLETED			

SYLLABUS

SECTIONS

1. **Graphical User Interface design**
2. **Controls**
3. **Visual Basic Environment**
4. **Coding**
5. **Monitoring and Responding to User Input**
6. **Graphical methods**
7. **Testing, Debugging and Error handling**
8. **Multiple-form applications**
9. **File Handling**

1. Graphical User Interface design

1.1 Guiding principles and standards
1.2 Windows GUI principles

2. Controls

2.1 Controls and their properties
2.2 Events and Methods
2.3 Focus and Tab order
2.6 Control arrays

3. Visual Basic environment

3.1 Find, Replace and Syntax checking

4. Coding

4.1 Assignment statements
4.2 Code intelligibility
4.3 Constants and Variables and Arrays
4.4 Operators
4.5 String manipulation
4.6 Control structures
4.7 Invoking control event procedures from code
4.8 Random numbers
4.9 General and function procedures

5. Monitoring and Responding to User Input

5.1 Message and Input boxes
5.2 Keyboard input
5.3 Selected text
5.4 Access letters
5.5 Drag and Drop

6. Graphical methods

6.1 Screen object and co-ordinate system
6.2 Drawing Lines and Circles

7. Testing, Debugging and Error handling

7.1 Testing and setting watch expressions
7.2 Error handling code in procedures

8. Multiple-form applications

8.1 Active control
8.2 Inter-form communication
8.3 Creating and using dialog boxes

9. File handling

9.1 Sequential files

Candidate's signature	Date

1. GRAPHICAL USER INTERFACE DESIGN

1.1 GUIDING PRINCIPLES & STANDARDS

1.1.1 Describe the 'Visual Processing Arc' and the guiding principles of good visual screen design, the use of colour on a screen and the guiding principles for the use of text fonts. (Appendix A) (W)

1.2 WINDOWS GUI PRINCIPLES

1.2.1 Describe the standards and main uses of:
the VGA colour scheme
assumed light source position in interface design
graphic lines on screen objects
text, graphic and control buttons.
(Appendix A) (W)

2. CONTROLS

2.1 CONTROLS & THEIR PROPERTIES

2.1.1 Use and describe the controls in Appendix B. (WP)

2.1.2 Use the control properties in Appendix B (P)

2.2 EVENTS AND METHODS

2.2.1 Write event handling code for the events listed in Appendix B. (P)

2.2.2 Use the methods given in Appendix B (P)

Candidate's signature	Date

2.3 FOCUS AND TAB ORDER

2.3.1 Describe the meaning of the term 'Focus', its effects on a control and the means by which a control can receive the 'Focus'. (W)

2.3.2 Set the 'Tab Index' of controls to preset the order in which they receive 'Focus'. (P)

2.4 CONTROL ARRAYS

2.4.1 Describe the purposes and functions of control arrays and the Index property. (W)

2.4.2 Create and use control arrays. (P)

3. VISUAL BASIC ENVIRONMENT

3.1 FIND, REPLACE AND SYNTAX CHECKING

3.1.1 Use the 'Find' and 'Find Next' menu commands to locate specified words and strings in a project. (P)

3.1.2 Use the 'Replace' menu command to replace specified words and strings throughout a project. (P)

3.1.3 Describe the functions of the Visual Basic syntax checker and the use of Option Explicit. (W)

3.1.4 Use Option Explicit. (P)

Candidate's signature	Date

4. CODING

4.1 ASSIGNMENT STATEMENTS

4.1.1 Describe the assignment statement syntax used to set or retrieve a control property or variable for:
the current Form or Module
another Form or Module
and the syntax used to set or retrieve the control property that is the value of the control. (W)

4.2 CODE INTELLIGIBILITY

4.2.1 Use consistent indentation and presentation of code to improve intelligibility. (P)

4.2.2 Write code in procedures and functions. (P)

4.3 CONSTANTS, VARIABLES AND ARRAYS

4.3.1 Use local and global symbolic constants, making use of constants given in the Visual Basic file CONSTANT.TXT. (P)

4.3.2 Describe the meaning of the terms 'data type' and 'data type mismatch' (Appendix B) (W)

4.3.3 Use Dim, Static and Global variables. (P)

4.3.4 Describe and state the scope and lifetime of Dim and Static variables declared in procedures and Dim and Global variables declared in modules. (Appendix B) (W)

4 3.5 Use arrays of integer and string variables. (P)

Candidate's signature	Date

4.4 OPERATORS

4.4.1 Use the logical operators listed in Appendix B. (P)

4.4.2 Use the relational operators listed in Appendix B. (P)

4.4.3 Use the precedence rules for arithmetic and logical operators. (P)

4.4.4 Describe the logical and relational operators, the precedence rules for arithmetic and the effects of parenthesis. (W)

4.5 STRING MANIPULATION

4.5.1 Use and describe the functions for concatenating and manipulating strings, listed in Appendix B. (WP)

4.6 CONTROL STRUCTURES

4.6.1 Use and describe the control structures and the DoEvents statement listed in Appendix B. (WP)

4.7 INVOKING CONTROL EVENT PROCEDURES FROM CODE

4.7.1 Invoke 'Click' event procedures for the Command Button and the Option Button, by setting the control's Value property to 'True'. (P)

4.8 RANDOM NUMBERS

4.8.1 Use and describe the 'Randomize' statement and the 'Rnd' function to generate random integer numbers. (WP)

Candidate's signature	Date

4.9 GENERAL AND FUNCTION PROCEDURES

4.9.1 Create and use general Sub Procedures in forms and in separate code modules. (P)

4.9.2 Create and use general Function Procedures in forms and in separate code modules. (P)

5. MONITORING & RESPONDING TO USER INPUT

5.1 MESSAGE AND INPUT BOXES

5.1.1 Use the 'MsgBox' statement and function to display a message to the user and to obtain a return value. (P)

5.1.2 Use the InputBox$ function to obtain an input string from the user. (P)

5.2 KEYBOARD INPUT

5.2.1 Describe the function of the 'KeyPreview' property. (W)

5.2.2 Use KeyPress and KeyDown events to monitor and acquire keyboard input from the user. (P)

5.2.3 Use the KeyAscii argument and the Chr$ function to echo user keyboard input. (P)

5.2.4 Use the Val and StrComp functions in the validation of numeric entries. (P)

5.3 SELECTED TEXT

5.3.1 Use the SelLength, SelStart and SelText properties to manipulate text in a Text box. (P)

Candidate's signature	Date

5.4 ACCESS LETTERS

5.4.1 Use and describe access letters in captions, menus and controls. (WP)

5.5 DRAG AND DROP

5.5.1 Use the Drag method and the DragDrop event to initiate actions. (P)

6. GRAPHICAL METHODS

6.1 SCREEN SYSTEM OBJECT & CO-ORDINATES

6.1.1 Use the properties of the Screen object to position forms on the screen. (P)

6.1.2 Use and describe the Visual Basic co-ordinate system as it applies to:
the screen
a form, and its location on the screen
a container, and its location on a form.
(Appendix C) (WP)

6.2 DRAWING LINES AND CIRCLES

6.2.1 Use the Line method to draw lines and boxes on forms. (P)

6.2.2 Use the Circle method to draw circles, ellipses and arcs on a form. (P)

Candidate's signature	Date

7. TESTING, DEBUGGING AND ERROR HANDLING

7.1 TESTING & SETTING WATCH EXPRESSIONS

7.1.1 Use Visual Basic 'Watch' expressions to halt code execution for specific conditions. (P)

7.1.2 Use the Immediate pane in break mode to determine the value of variables and expressions. (P)

7.1.3 Identify and correct simple errors in program code. (P)

7.2 ERROR HANDLING CODE IN PROCEDURES

7.2.1 Describe;
- simple sources of run-time errors
- responses by Visual Basic when an error occurs
- simple uses of the OnError statement
- use of GoTo and Resume to manage errors
- use of Err and Error$ in error-handling (W)

7.2.2 Create simple error-handling routines. (P)

7.2.3 Use the Error statement to test a Visual Basic error-handling routine. (P)

8. MULTIPLE-FORM APPLICATIONS

8.1 ACTIVE CONTROL

8.1.1 Use and describe the Show and Hide methods and the Load and Unload statements. (WP)

8.1.2 Use the ActiveForm and ActiveControl properties in applications having more than one form. (P)

Candidate's signature	Date

8.2 INTER-FORM COMMUNICATION

8.2.1 Create simple applications in which event procedures on one form reference properties on another form. (P)

8.2.2 Create simple multi-form applications that use procedures in a code module. (P)

8.3 CREATING AND USING DIALOG BOXES

8.3.1 Describe the use of Dialog boxes and the main design features of forms used as Dialog boxes. (W)

8.3.2 Design simple Dialog boxes that include Command buttons using the Cancel and Default properties. (P)

8.3.3 Create simple applications using menu controls to call up Dialog boxes and position them. (P)

8.3.4 Acquire and use information input by the user to a modal Dialog box to modify the state of the form calling up the Dialog and/or its controls. (P)

8.3.5 Use the Common Dialog control to provide access to the standard set of dialog boxes listed in Appendix B. (P)

9 FILE HANDLING

9.1 SEQUENTIAL FILES

9.1.1 Use the FreeFile function to obtain the next valid, unused file number. (P)

9.1.2 Open a file for sequential output and write text box contents to the file. (P)

9.1.3 Open a file for sequential input and read the contents of the file into a string variable. (P)

9.1.4 Use the Close statement to close an open file. (P)

Candidates Instructions

This test consists of 19 multiple-choice questions. In order to pass you must answer a minimum of 13 of them correctly. You have one **hour** to complete the test.

The first number for each question is the test question number. This is the number which you should use on the answer sheet. The number in brackets relates to your log book reference.

1 (1.1.1.1)

Which one of the following statements about the principles of screen design is **FALSE**?

a The eye tends to follow a curve from top-left to bottom-right on the screen.
b Black and white objects stand out from a group.
c Isolated elements are more attractive to the eye than groups.
d Graphics are more attractive than text.

2 (1.2.1.1)

Good Windows design

a spaces buttons as widely apart as possible
b uses a variety of fonts
c puts all the command buttons in one group
d uses standard button sizes.

3 (2.1.1.1)

Which one of the following is **FALSE**?

a Each menu can have up to 4 levels of submenus.
b A menu access key can be created by preceding the selected letter with **&**.
c Check marks on a menu indicate the state of the enabled property.
d A menu control can be made invisible to make it unavailable to the user.

4 (2.3.1.1)

The active control is the one which

a is enabled
b is visible
c has the focus
d has its tab index=0.

5 (2.4.1.1)

Which one of the following is an advantage of using a control array?

a a single identifier for the set of controls
b simplified function procedure code
c a separate tab index for each element
d each control has its own event procedure

6 (3.1.3.1)

The Option Explicit statement is used to

a prevent the typing of undeclared variables at design time
b make the compiler automatically alter misspelt variable names
c avoid the use of declaration statements
d force explicit declaration of all variables.

7 (4.1.1.1)

Which one of these statements will change the interval property of a timer control using a scroll bar control?

a **tmrTimer=hsbInterval.Value**
b **tmrTimer.Interval=hsbInterval.Scroll**
c **tmrTimer.Value=hsbInterval**
d **tmrTimer.Interval=hsbInterval.Value**

8 (4.3.2.1)

Which one of the following is not a valid data type in Visual Basic?

a Logical
b Long
c String
d Variant

9 (4.3.4.1)

Which one of the following statements is **FALSE**?

a Making a local variable *Static* preserves its value while the application is running.
b *global* variables must be declared in a code module.
c A *Dim* statement can only be made in the declarations area of a form.
d The scope of a variable is the range of statements for which the variable is valid.

Values	
A	**B**
0	0
1	2
2	4
3	6
4	8

Fig 4.4.4.1

10 (4.4.4.1)

Which one of the following expressions will evaluate to TRUE for 4 of the pairs of values of A and B in the table in figure 4.4.4.1 ?

a A<2 **AND** B>4
b A<2 **OR** B>4
c A<3 **AND** B>2
d A<3 **OR** B>2

11 (4.5.1.1)

Which one of the following is **FALSE**?

a **RIGHT$** ("VISUAL",2) = "AL"
b **MID$** ("VISUAL",2,2) = "SU"
c **LEN** ("VISUAL") = 6
d **INSTR** ("VISUAL","S") =3

12 (4.6.1.1)

A pre-condition loop

a is performed at least once
b may not be performed at all
c does not need an end-of-loop indicator
d increments a counter when the loop starts

13 (4.8.1.1)

Which one of the following statements is **FALSE**?

a The **RND** function returns a value between 0 and 1
b The **RND** function returns a value between +32,767 and -32,768.
c The **RND** function produces the same sequence of random numbers unless **RANDOMIZE** is used.
d The **RANDOMIZE** statement uses the Timer function to return a seed value if no argument is used.

14 (5.2.1.1)

When the **KeyPreview** property of a form is set to true

a the application only allows a preview of Function keys
b only the control with the focus can receive a keyboard event
c the application allows handling of only the **KeyPress** event
d the form receives all keyboard events before other form controls.

15 (5.4.1.1)

Access letters on captions, menus and controls are used by pressing

a Ctrl+Alt+Access letter
b Alt+Access letter
c Ctrl+Access letter
d Ctrl+SHIFT+Access letter

16 (6.1.2.1)

Which one of the following is **NOT** true?

a The resolution of the screen depends on the number of pixels per inch.
b The default **ScaleMode** unit is inches.
c There are 1,440 twips to an inch.
d The **SCALE** method is used to define the coordinate system.

17 (7.2.1.1)

Which one of the following statements is **TRUE**?

a The **ERR** function prints an error message if an error has occurred.
b The **ERR** function returns a variant corresponding to a given error condition.
c The **ERROR$** function returns a string defining the most recent run time error.
d The **ERROR$(ERR)** function simulates the occurrence of an error.

18 (8.1.1.1)

The **SHOW** method is used to

a display the startup form
b display the specified form
c display controls that have the Visible property set to False
d bring a form to the front of the Z-order

19 (8.3.1.1)

When a modal dialog is shown

a the next line of code is not executed until the dialog is closed
b it must have two buttons with the Default and Cancel properties set to True
c the form Load event must contain code to position the form
d keyboard and mouse input to the containing form is recognised as usual.

Candidates Instructions

This test consists of 19 multiple-choice questions. In order to pass you must answer a minimum of 13 of them correctly. You have one **hour** to complete the test.

The first number for each question is the test question number. This is the number which you should use on the answer sheet. The number in brackets relates to your log book reference.

1 (1.1.1.2)

The visual processing arc followed by the eye when initially scanning a VDU screen goes from

a top left to bottom right
b top right to bottom left
c bottom right to top left
d bottom left to top right.

2 (1.2.1.2)

Which one of the following sets of panel edge colours creates the visual effect of a raised panel in the standard GUI?

a top & left white, bottom & right dark grey
b top & bottom white, left & right dark grey
c bottom & right white, left & top dark grey
d left & right white, top & bottom dark grey

3 (2.1.1.2)

Which one of the following is **NOT** a function of a window control menu box?

a Maximise
b Scale
c Close
d Move

4 (2.3.1.2)

In a windows GUI the active control is the

a control on which the cursor is positioned
b last control clicked with the mouse
c control which is enabled
d control with focus.

5 (2.4.1.2)

Which one of the following is a reason for organising a group of controls of identical type as a control array?

a Reduction of interface development time.
b Use of a single event procedure for group.
c Enables event procedures to be copied.
d Enables same function to be created on a number of forms.

6 (3.1.3.2)

When Option Explicit is **NOT** used, the data type of undeclared variables

a is variant
b depends on the variable
c can only be string or integer
d is automatically set to Any.

7 (4.1.1.2)

Which one of the following is the assignment statement to transfer the Text property of a text box control into the string variable **temp$**

a **Text=txtBox.temp$**
b **temp$=Text.txtBox**
c **temp$=txtBox.Text**
d **txtBox.Text=temp$**

8 (4.3.2.2)

Which one of the following would generate a "type mismatch" error?

a X$ = "25"
b X$ = "B"
c X$ = K%
d X$ = "K%"

9 (4.3.4.2)

The scope of a variable declared in a sub procedure with the Dim statement is

a changed when Option Explicit is used
b global to the project
c local to the module containing the sub procedure
d local to the sub procedure

Values	
A	**B**
0	0
1	2
2	4
3	6
4	8

Fig 4.4.4.1

10 (4.4.4.2)

Which one of the following expressions will evaluate to TRUE for 4 of the pairs of values of A and B in the table in figure 4.4.4.1 ?

a A<=2 **AND** B>4
b A<2 **OR** B>=4
c A>=1 **AND** B>=2
d A<=1 **OR** B<=2

11 (4.5.1.2)

If x$ = "PQRST" then **MID$**(x$,2,1) returns

a "P"
b "Q"
c "PQ"
d "QR"

```
Select Case X
Case Is > 2
     Debug.Print "X is > 2"
Case Is > 4
     Debug.Print "X is > 4"
Case Is > 6
     Debug.Print "X is > 6"
Case Else
     Debug.Print "Other value"
End Select
```

Figure 4.6.1.2

12 (4.6.1.2)

Which one of the following lines is printed by the Select Case statement shown in figure 4.6.1.2 when the floating point variable X contains the value 4.761.

a `X is > 2`
b `X is > 4`
c `X is > 6`
d `Other value`

13 (4.8.1.2)

The Randomize statement

a generates a random integer
b generates a floating point number in the range 0 to 1
c seeds the random number generator
d requires a number argument.

14 (5.2.1.2)

When the **KeyPreview** property of a form is set to False

a the active control receives Keyboard events
b the form receives Keyboard events after the active control
c only the KeyUp and KeyDown events are received by the form
d Keyboard events are not received by the form or its controls.

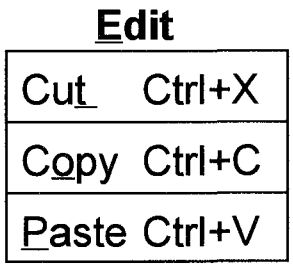


Figure 5.4.1.2

15 (5.4.1.2)

Which one of the following key sequences will open the Edit menu shown in figure 5.4.1.2 and invoke the Copy command?

a Alt+E Ctrl+C
b Ctrl+E Ctrl+O
c Alt+E O
d Ctrl+C

16 (6.1.2.2)

A command button named **cmdButton1** is contained by a frame that is located in a custom dialog box. When the dialog box is showing, the assignment **X=cmdButton1.Top** will set the value of X to the distance from the top of the button to the top of the

a picture
b dialog box
c form
d screen.

17 (7.2.1.2)

The main use of the GoTo statement in Visual Basic is to

a exit a sub routine when an error occurs
b run code in general sub procedures
c run an error handler when an error occurs
d enable sections of code to be repeated.

18 (8.1.1.2)

The form **frmForm1** is hidden using the Hide method followed by execution of the statement **X%=frmForm1.Visible**.

Which one of the following is the state of the variable X

a False
b True
c Null
d 1

19 (8.3.1.2)

When a form is used as a custom dialog box it

a is usually iconised when it is closed
b can be modal or modeless
c must contain a Cancel button
d cannot have its own menus.

Candidates Instructions

This test consists of 19 multiple-choice questions. In order to pass you must answer a minimum of 13 of them correctly. You have one **hour** to complete the test.

The first number for each question is the test question number. This is the number which you should use on the answer sheet. The number in brackets relates to your log book reference.

1 (1.1.1.3)

A bold font is recommended when

a text is in italics
b small fonts are used
c text needs to be legible when 'greyed'
d a serif font is used.

2 (1.2.1.3)

Which one of the following statements is **FALSE**?

a Disabled buttons are made blank.
b Text buttons have a text caption.
c Graphic buttons have a picture.
d Control buttons have a functional symbol.

3 (2.1.1.3)

Which one of the following controls enables the user to select from a colour palette at run time?

a Combo Box.
b Common Dialog.
c Picture Box.
d Image Box.

4 (2.3.1.3)

Which one of the following controls is capable of receiving focus?

a Form.
b Label.
c Shape.
d Timer.

5 (2.4.1.3)

The index property of a control array is used to

a identify which array element has the focus
b fix the control position in the object list
c reference each individual control
d set the Tab order.

6 (3.1.3.3)

When the Option Explicit statement is used in a Visual Basic project, undeclared variables

a are limited to local scope
b can only be of the Variant data type
c must be initialised before they can be used
d generate error messages at run time.

7 (4.1.1.3)

Which one of the following **CANNOT** set the Caption of a form to 'profits'?

a **Caption="profits"**
b **Me.Caption="profits"**
c **Form1.Caption="profits"**
d **frmProfits.Caption=profits&**

8 (4.3.2.3)

Which one of the following statements will produce a 'type mismatch' error?

a n$=Me.Caption
b x$=Val(n$)
c Me.FontName="Arial"
d I%=Abs(-5)

9 (4.3.4.3)

A constant declared with the keyword Global in a module can

a be declared with a different value inside a sub procedure
b be accessed only in sub procedures in modules
c has global scope only in the module within which it is declared
d be used wherever a variable of the same data type can be used.

10 (4.4.4.3)

Which one of the following statements is **TRUE**?

a 3+2*2<5-2*2
b 5-4/2>1+6/2
c 9-3*3<7-3*2
d 4+2*3>3+4*2

11 (4.5.1.3)

The Asc("A") function returns

a a null terminated string
b the Key board event code for the letter 'A'
c the ANSI code for the letter 'A'
d the character code for the letter 'A'.

12 (4.6.1.3)

The **DoEvents** statement

a allows other applications to interrupt the Visual Basic application
b stops the current application
c cycles through all event procedures on the active form
d relinquishes time to other applications.

13 (4.8.1.3)

The statement Int(6*Rnd+1) generates random numbers in the range

a 0 to 5
b 0 to 7
c 1 to 6
d 6 to infinity.

14 (5.2.1.3)

A reason for setting the **KeyPreview** property of a form to True is to

a control the KeyAscii keycode passed to the active control
b stop keyboard events being passed by the active control
c trap the Ctrl, Shift and Alt keys
d echo keyboard input to the active control.

15 (5.4.1.3)

The caption property of a menu option is set to **Co&ntents**. The access key for this menu option is

a O
b Alt+O
c N
d Alt+N

16 (6.1.2.3)

Which one of the following sets of properties defines the dimensions of the client area of a form?

a Height and Width.
b ScaleHeight and ScaleWidth.
c ScaleLeft and ScaleTop.
d Screen.Height and ScreenWidth.

17 (7.2.1.3)

Which one of the following will cause a run time error to occur in a Visual Basic program?

a A call to a general sub procedure containing a syntax error.
b A reference to the value of a local variable that has not been initialised.
c A reference to a property of a form that has not been loaded.
d A reference to an out of range array subscript.

18 (8.1.1.3)

When a form is unloaded

a it is removed from memory
b it is hidden but remains in memory
c focus is removed to form main
d the QueryUnload event must be used to reset the form's default properties.

19 (8.3.1.3)

Which one of the following is **TRUE** when a dialog box contains a single command button with the caption 'OK' and its Default property set to True?

a The dialog box will close when the Enter key is pressed.
b The button's Click event is invoked when the Enter key is pressed.
c The dialog box will close when the Esc key is pressed.
d The button's Click event is invoked when the Esc key is pressed.

Candidates Instructions

This test consists of 19 multiple-choice questions. In order to pass you must answer a minimum of 13 of them correctly. You have one **hour** to complete the test.

The first number for each question is the test question number. This is the number which you should use on the answer sheet. The number in brackets relates to your log book reference.

1 (1.1.1.4)

Which one of the following is best **avoided** when designing a graphical user interface?

a Using large areas of bright colour.
b Using blue as a background colour.
c Using colour to indicate grouping.
d Using similar colours close together.

2 (1.2.1.4)

A GUI control button displaying a text caption is MOST likely to be used in a

a toolbar
b ruler
c dialog box
d status bar.

3 (2.1.1.4)

Which one of the following controls would be best for displaying and carrying out arithmetic operations on tabulated data?

a Combo box.
b Grid.
c Form.
d List box.

4 (2.3.1.4)

A command button named **cmdExit** is on the active form. For which set of the following properties of the button will the instruction **cmdExit.SetFocus** be successful.

a Default and Cancel are NOT both True
b Enabled and TabIndex are both True
c Visible and TabStop are both True
d Visible and Enabled are both True.

5 (2.4.1.4)

The instruction **optBlack(2).Value=1** refers to the

a option button with the name optBlack(2)
b first two option buttons in a control array
c second option button in a control array
d third option button in a control array.

6 (3.1.3.4)

A function of the Visual Basic syntax checker is to ensure that

a all variables are declared and initialised
b the meaning of code is clear
c statements are grammatically correct
d statements are logically consistent.

7 (4.1.1.4)

Which one of the following statements could be used to determine the value of an Option button control?

a **Value.bVal%=optButton1**
b **bVal%=optButton1.Value**
c **optButton1=bVal%**
d **bVal%=Value(optButton1)**

8 (4.3.2.4)

A 'data type mismatch' error will occur when

a a variable is declared with the wrong type
b the Val function is applied to a String type variable
c the Int function is applied to a String type variable
d an integer value is assigned to a Single type variable.

9 (4.3.4.4)

The 'lifetime' of a local variable is

a the time to execute the procedure in which it is declared
b the time after it is initialised
c the extent to which it is available to code in the project
d less for Static than for Dim declarations.

10 (4.4.4.4)

The expression 10-6/2+6*(7-2*3) evaluates to

a 8
b 13
c 88
d 97

11 (4.5.1.4)

The function Trim$

a rounds decimal values to nearest integer
b removes fractional part of decimal value
c removes all spaces from a string
d removes leading and trailing spaces from a string.

```
n=5
      Do Until n>10
            Debug.Print "loop"
            n=n+1
      Loop
```

Figure 4.6.1.4

12 (4.6.1.4)

How many times does the code shown in figure 4.6.1.4 print the word "loop"?

a 5
b 6
c 10
d 11

13 (4.8.1.4)

What are the minimum and maximum values generated by the expression **Int(6*Rnd+1)**

a 0 and 1
b 0 and 6
c 1 and 6
d 6 and 7

14 (5.2.1.4)

Which one of the following statements is **TRUE**?

a KeyDown, KeyUp and KeyPress events can only be received by text boxes and controls with an access letter set.
b Keyboard input generates the SendKeys event for the active control.
c The state of the KeyPreview property determines whether or not the form receives keyboard events.
d The state of the Ctrl, Alt and Shift keys determine which controls receive keyboard events.

15 (5.4.1.4)

If a command button has its caption set to "E&xit" then the button's Click event is invoked when the user presses the key combination

a Tab + X
b Shift + X
c Ctrl+X
d Alt+X

16 (6.1.2.4)

A MouseDown event occurs on a control. With the mouse button still held down the user moves the mouse pointer off the control and then releases the mouse button. The MouseUp event will

a be lost
b occur as pointer moves back over control
c occur on the control over which the button was released
d occur on same control as MouseDown.

17 (7.2.1.4)

The **Resume Next** statement is used at the end of an error-handling routine, to resume program execution with the statement

a that caused the error
b immediately following the error
c in the first sub procedure after the error
d at the start of the error handling routine.

18 (8.1.1.4)

All property values set on a form and its controls at run time are returned to the default design time settings when the form is

a hidden
b refreshed using the Refresh method
c reduced to an icon
d unloaded.

19 (8.3.1.4)

Which one of the following purposes is a custom dialog box **MOST** suitable for?

a Warning users of outcomes of an action.
b Requesting user string or number entry.
c Offering users choices of settings-options.
d Offering users standard functions.

PRACTICAL ASSIGNMENT: DEBUG AND DEVELOP A SIMPLE DRAWING PROGRAM BASED ON THE CIRCLE METHOD

1. OBJECTIVE REFERENCES

2.1.1, 2.1.2, 2.2.1, 2.2.2, 2.3.2, 2.4.2, 3.1.1, 3.1.2, 4.2.1, 4.2.2, 4.3.1, 4.4.1, 4.4.2, 5.2.4, 6.1.1, 6.1.2, 6.2.2, 7.1.1, 7.1.2, 7.1.3, 8.3.2, 8.3.3, 8.3.4, 8.3.5

2. PREPARATION

2.1	Location of test	The training centre or other venue where supervision and appropriate working conditions will be provided.
2.2	Requirements	Microcomputer or Work station with VGA or SVGA colour monitor running Windows GUI version 3.0 or later version and Visual Basic version 3.0 or later version. User manual for Visual Basic programming language. Copy of para 6.1 Copy of **.mak** file and supporting files on disk.
2.3	Tutor notes	The tutor must provide the candidate with a copy of the project **draw.mak** (Code is provided in appendix D, electronic files are also available, see note 2 on page Intro/1 of this module) comprising **draw.frm**, **drawdlg.frm** and **cmdialog.vbx**. The candidate must also be provided with the action chart for the program (para 6.1) and associated diagrams and form images in para 6.2. This assignment may be taken over more than one session. Tutors must ensure that all candidate materials are collected at the end of each session and that the work presented represents only that produced by the individual candidate in the time allocated. The tutor must demonstrate a working version of **draw.mak** and **draw_1.mak** that has the additional facility to draw arcs, to ensure that candidates understand its operation.

3. CANDIDATE'S INSTRUCTIONS

Part 1

3.1 This (complete) assignment must be completed within 4 hours. You are advised to read all of the candidate's instructions before commencing work.

You have been provided with the project **draw.mak** comprising the files **draw.frm**, **drawdlg.frm** and **cmdialog.vbx**, together with an action chart for the program.

The function of the program is to draw filled circles, ellipses, pie slices and arcs in response to data entered by the user, using the Visual Basic **Circle** method. The operation of the required program will be demonstrated to you.

Study the action chart and ensure that you are familiar with the structure and operation of the program.

3.2 Open the project **draw.mak**. Locate the declaration of the constant *CLEAR_CIRCLE* in the declarations section of **frmDraw**. The value of this constant is the index value of the **Erase** command button on the form. The name chosen for this constant does not indicate its purpose very well. Use the Visual Basic Find and Replace facility to change the name of this constant throughout the project from *CLEAR_CIRCLE* to *ERASE_DRAWINGS*.

3.3 A section of the code in the **cmdDraw_Click** sub procedure on **frmDrawDlg** requires indenting to make its operation clear. Locate this section of code and indent it.

3.4 The code in **frmDraw** and **frmDrawDlg** contains a number of coding errors that must be corrected (debugged) before the program will operate satisfactorily. By attempting to run the project **draw.mak** you will note that Visual Basic reports a Type Mismatch error in the **frmDraw_Load** event procedure. Correct this error.

3.5 When you have corrected the mismatch error, run the program and attempt to draw a circle. Visual Basic will go into Break mode immediately following the drawing operation and will report an Illegal Function Call error in the **cmdTask_Click** event sub procedure on **frmDraw**. Modify the sub procedure code to eliminate this error.

3.6 Attempted drawing of the other shapes will also result in errors. Eliminate all of these errors.

3.7 Run the program and attempt to draw an ellipse. The program should run without any errors being reported, but an ellipse will not be drawn. Set one or more breakpoints in the code and use the Visual Basic debugging facilities to trace program execution. Locate and correct the coding error that prevents an ellipse being drawn.

3.8 There is a known problem with the operation of this program. When a large shape is drawn and is then subsequently partly covered by a dialog box, the covered part is not redrawn when the dialog box closes. Change one of the **frmDraw** property values, either in the program code or in the Properties box, to eliminate the problem.

3.9 The version of the program supplied to you lacks the ability to draw arcs. Add an additional command button to the control array on **frmDraw**. This button will take an index value of 5.

3.10 Change the caption on the button that currently has the caption **Erase** to give it the caption **Arc**. Change the caption on the button that currently has the caption **Exit** to give it the caption **Erase**. Set the caption on the new button to **Exit**.

3.11 Use the VB *Find* facility to locate the declarations of constants *ERASE_DRAWINGS* and *EXIT_PROGRAM*. Modify the value of the *ERASE_DRAWINGS* and *EXIT_PROGRAM* constants to match the Index values of the new **Erase** and **Exit** buttons. Declare an additional constant *DRAW_ARC* and set its value to the Index value of the Arc button.

3.12 Add to the code in the **Load** event sub procedure of **frmDraw** to show the Arc button with the group of drawing buttons. Modify the existing code so that the set of buttons are positioned approximately centrally on the screen. Run the program to ensure that the buttons appear as required.

3.13 Add additional code to the project as necessary to enable an arc to be drawn (a section of the circumference of a circle). The data entry dialog box must appear with the caption "**Draw Arc**". The labels and text boxes for **Aspect** must not be shown, nor the label and text box for **Fill Colour** because an open shape cannot be filled.

Take careful note of the comments which appear at the end of the **cmdDraw_Click** sub procedure on **frmDrawDlg**.

3.14 Test that an arc is correctly drawn for valid data.

3.15 Place comments in the program that contain your name and the date. Save your work to a diskette with the name **DRAW_1.MAK**. Make sure that your name, the project name and the date are written on the diskette label and hand them to your tutor.

4. MARKING

4.1	Completed within four hours.	[ ]
4.2	Constant CLEAR_CIRCLE changed to ERASE_DRAWINGS throughout.	( )
4.3	Indenting correctly implemented in **cmdDraw_Click** sub procedure.	( )
4.4	Type Mismatch error in **frmDraw_Load** corrected	[ ]
4.5	Illegal Function Call error corrected in Circle draw.	[ ]
4.6	Illegal Function Call errors when drawing other shapes corrected.	[ ]
4.7	Errors in drawing an ellipse corrected.	[ ]
4.8	Shape is re-drawn correctly after clearing a dialogue box.	( )
4.9	Additional button added as an element of **cmdDraw()** control array.	[ ]
4.10	Button captions changed correctly.	[ ]
4.11	Values of form constants match button functions and Arc constant added.	[ ]
4.12	Arc button and button group correctly positioned.	( )
4.13	Clicking 'Arc' button gives correct entry dialog box with required labels.	[ ]
4.14	Arc correctly drawn for valid data.	[ ]
4.15	Project saved to diskette and printed copy provided.	[ ]

5. ASSIGNMENT COMPLETION

The candidate will have satisfactorily completed this assignment if successful in all items marked with a [ ] and at least two of the items marked with a ( ).

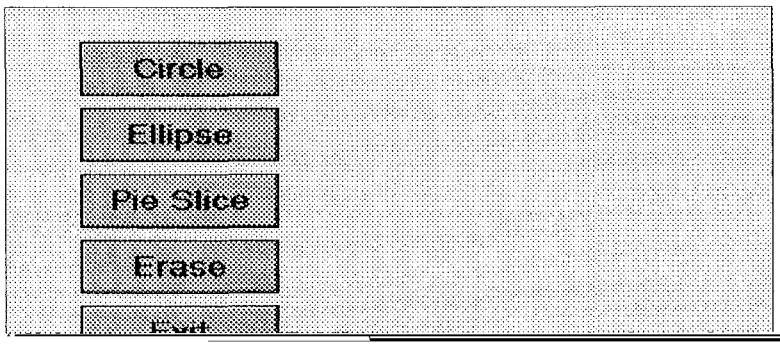
A period of at least seven days must elapse before an unsuccessful candidate may retake this assignment.

6 ASSIGNMENT DOCUMENTATION

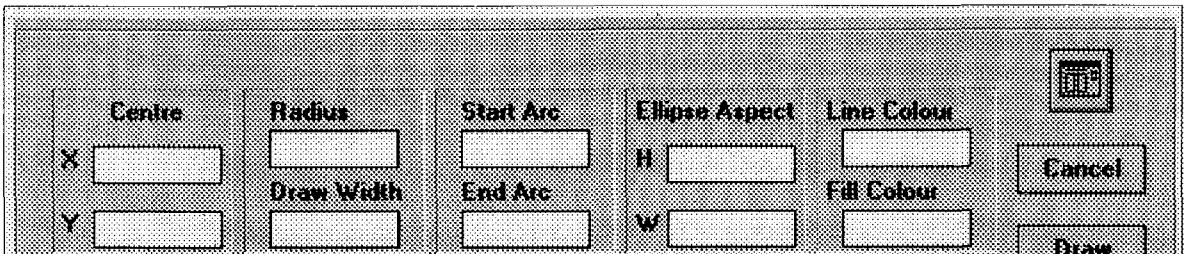
6.1 Action chart for DRAW.MAK

Form	Control	Sub or Function Procedure	Actions	Sub or Function Call
frmDraw	Form	Load	Maximise window size. Size and position command buttons	
frmDraw	Form	MouseDown	Copy mouse pointer X and Y coordinates to frmDrawDlg text boxes (txtCentreX and txtCentreY)	
frmDraw	cmdTask(0)	Click	Initialise drawing data dialog box, frmDrawDlg, for CIRCLE drawing and call the dialog	ShowDialog "Draw Circle"
frmDraw	cmdTask(1)	Click	Initialise drawing data dialog box, frmDrawDlg, for ELLIPSE drawing and call the dialog	ShowDialog "Draw Ellipse"
frmDraw	cmdTask(2)	Click	Initialise drawing data dialog box, frmDrawDlg, for PIE SLICE drawing and call the dialog	ShowDialog "Draw Pie Slice"
frmDraw	cmdTask(3)	Click	Erase all drawings using the Cls method	
frmDraw	cmdTask(4)	Click	Unload forms to close the program	
frmDraw	Form (general)	ShowDialog	Position and show modal frmDrawDlg dialog box with caption passed as argument	
frmDrawDlg	cmdCancel	Click	Hide frmDrawDlg	
frmDrawDlg	cmdDraw	Click	Convert data strings entered by user into frmDrawDlg text boxes into numbers and copy to local variables Validate the data: Invalid data - show message box and return user to dialog Valid data - hide dialog and draw chosen shape	
frmDrawDlg	txtFillColor	DblClick	Convert colour value returned by GetColor function to string and copy to txtFillColor	GetColor()
frmDrawDlg	txtLineColor	DblClick	Convert colour value returned by GetColor function to string and copy to txtLineColor	GetColor()
frmDrawDlg	Form (general)	GetColor	Show the Common Dialog Color box and get a colour value from the user	

6.2 Screen images of **frmDraw** and **frmDrawDlg**.



Note: Command buttons on **frmDraw** are shown in design-time positions. The buttons are re-positioned in the form **Load** event.



PRACTICAL ASSIGNMENT: CALCULATOR DESIGN

1. OBJECTIVE REFERENCES

2.1.1, 2.1.2, 2.2.1, 2.4.2, 3.1.4, 4.2.1, 4.2.2, 4.3.1, 4.3.3, 4.4.2, 4.4.3, 4.6.1, 4.7.1, 4.8.1, 5.1.1, 5.1.2, 5.4.1, 6.1.1, 6.2.1, 7.1.3, 8.1.1, 8.2.1, 8.2.2, 8.3.3, 8.3.4

2. PREPARATION

2.1	Location of test	The training centre or other venue where supervision and appropriate working conditions will be provided.
2.2	Requirements	Microcomputer or Work station with VGA or SVGA colour monitor running Windows GUI version 3.0 or later version and Visual Basic version 3.0 or later version. User manual for Visual Basic programming language. Copy of para 6 Copy of **.mak** file and supporting files on disk.
2.3	Tutor notes	The tutor must provide the candidate with a copy of the project **calcultr.mak** (The code is provided in appendix D and electronic files are available, see note 2 on page Intro/1 of this module) comprising **calcultr.frm**, **operator.frm** and **level.frm**. This assignment may be taken over more than one session. Tutors must ensure that all candidate materials are collected at the end of each session and that the work presented represents only that produced by the individual candidate in the time allocated. Network systems may be used, but care must be taken that candidates have no access to their electronic files outside the working sessions. If the machine does not have floppy disk copy facilities the tutor must inform the candidates how they must save their work for marking.

3. CANDIDATE'S INSTRUCTIONS

3.1 This assignment must be completed within four hours. You are advised to read all of the candidate's instructions before commencing work.

This assignment involves the creation of a Visual Basic program to meet the requirements of the following specification:

Your program will provide users with a calculator tool which will enable them to practice the four basic arithmetic operations of addition, subtraction, multiplication and division.

You will have been provided with a project file **calcultr.mak** that contains the three forms required by the assignment, **frmCalculator**, **frmOperator** and **frmLevel**, already constructed. The form images are shown in para 6.

When the project is run, **frmCalculator** must be permanently displayed (this is **Form1**) while the two forms **frmOperator** and **frmLevel** must be made to operate as custom dialog boxes called up from a menu system on **frmCalculator**. The assignment also requires the use of the MsgBox statement and the InputBox function to communicate with the user.

The project as provided does not include any code. The instructions below provide guidance as to how the project may be coded, but you are free to code the project as you see fit to achieve the required functions. The functions of the three project windows are as follows:

frmCalculator The calculation and its result are shown in Label controls. A typical calculation will appear as:

11	÷	5	=	2.2

Labels are also used to display the level of difficulty of the calculation and to show a message about accuracy when the chosen arithmetic operation is division.

frmLevel This dialog allows the user to set a level of difficulty (1, 2 or 3) for the calculation by clicking on one of three option buttons. The higher the level of difficulty the larger the number in the calculation.

frmOperator This dialog allows the user to choose the arithmetic operator (+, -, x or ÷) to be used in the calculation by clicking one of four option buttons.

In addition, the **InputBox** function is used to get an answer from the user, and the **MsgBox** statement is used to pass information to the user.

The **frmLevel** dialog has four lines drawn at design time that create the effect of a raised panel around the option buttons. An identical panel must be created on **frmOperator** when the dialog loads using the **Line** Method.

A menu system must be created on **frmCalculator** to enable the user to choose an action. The menu system must offer the following options:

- Present a new calculation
- Set the arithmetic operator for the calculation
- Set the level of difficulty for the calculation
- Display the score of correct answers in the current session
- Exit the program

When a new calculation is requested two random integers must be generated and displayed in **lblFirstNo** and **lblSecondNo** on **frmCalculator**. When a new arithmetic operator is set it must be displayed in **lblOperator**. When a new level of difficulty is set it must be displayed in **lblDiffLevel**. When the program starts the arithmetic operator must be set to + (addition) and the level of difficulty must be set to 1 by default. When ÷ (division) is the chosen operator **lblDivideMsg** must be made visible to inform the user that answers must be given to an accuracy of one decimal place, it must be hidden for all other operators.

Answers to calculations given by the user are to be entered via an InputBox, and this must be displayed when a new calculation is set. For a wrong answer an appropriate message must be shown in a MsgBox and then the user must be allowed to make another attempt. The user must be allowed three attempts to give the correct answer, after which the InputBox must be removed and the correct answer shown in **lblAnswer**. When the user gives the correct answer an appropriate message must be shown in a MsgBox and the answer shown in **lblAnswer**.

While the program is running a count must be kept of the number of calculations set and a count of the number of correct answers entered by the user.

3.2 Create the menu system for **frmCalculator** according to the following specification:

Menu Item	**Menu Name**
Calculator	mnuCalc
New Problem	mnuCalc_New
Show Score	mnuCalc_Score
Options	mnuOptions
Operator Type	mnuOptions_Operator
Difficulty Level	mnuOptions_Level
Exit	mnuExit

3.3 Create a BAS module and save it with the project as **calcultr.bas**.

3.4 Declare the following global variables as integer type in the **.bas** module:

Operator
DifficultyLevel

The variable 'Operator' is to be used to store the arithmetic operator selected by the user as a value between 0 and 3; these values correspond to the Index values of the **optOperator** option buttons on **frmOperator**.

The variable 'DifficultyLevel' is to be used to store the level of difficulty selected by the user as a value between 0 and 2; these values correspond to the Index values of the **optLevel** option buttons on **frmOperator**.

3.5 Declare the following global constants in the .**bas** module:

These values correspond to the Index values of the **optOperator** buttons.

```
ADD = 0
SUBTRACT = 1
MULTIPLY = 2
DIVIDE = 3
```

These are string constants for the arithmetic operator symbols to be displayed in the calculation (See note * below).

```
ADD_SYMBOL = "+"
SUBTRACT_SYMBOL = "-"
MULTIPLY_SYMBOL = "x"
DIVIDE_SYMBOL = "÷"
```

These values correspond to the Index values of the **optLevel** buttons.

```
LEVEL_1 = 0
LEVEL_2 = 1
LEVEL_3 = 2
```

* Note: The multiplication and division symbols are not available on the standard keyboard but they can be entered in the following way: to enter the 'x' symbol hold down the Alt key and type 0215, then release the Alt key; to enter '÷' hold down the Alt key and type 0247.

3.6 In the general declarations section of **frmCalculator** declare the following variables:

Number1 as Integer type (first number used in the calculation)
Number2 as Integer type (second number used in the calculation)
UserAnswer as Single type
CorrectAnswer as Single type
CorrectAnsCount as Integer type
QuestionCount as Integer type
Attempts as Integer type
InputBoxTop as Single type
InputBoxLeft as Single type

3.7 Write code in the **frmLoad** event sub procedure of **frmCalculator** to carry out the following tasks:

- Initialise the Operator and DifficultyLevel variables.
- Initialise the label controls to display default captions.
- Initialise the **InputBoxTop** and **InputBoxLeft** variables to suitable values that can be used when the **InputBox** is shown to position the box below **frmCalculator**, and roughly centred horizontally.
- Position **frmCalculator** near the top of the screen and centred horizontally.

3.8 Write code in the **mnuCalc_New_Click** event sub procedure of **frmCalculator** to carry out the following tasks:

- Generate two random numbers for the calculation to suit the level of difficulty.
  For level 1 the numbers to be in the range 1 - 10
  For level 2 the numbers to be in the range 5 - 20
  For level 3 the numbers to be in the range 21 - 99
- Update the form labels to display the next calculation, making sure that:
  a) if the operator is subtraction then the first number in the calculation is the larger of the two to ensure a positive answer;
  b) if the operator is division then the label requesting answers to one decimal place accuracy is shown.
- Calculate the correct answer to the calculation and store the result. If the operator is division then the result should be formatted to one decimal place.
- Get the user's answer to the calculation using the **InputBox** function. Show an appropriate message using the **MsgBox** statement for a correct answer and for an incorrect answer. Limit the number of attempts to answer the question to three.
- Update the count of questions and of correct answers.
- Return the program to the ready state if the user cancels the InputBox.

3.9 Write code in the **mnuCalc_Score_Click** event sub procedure of **frmCalculator** to display a message showing the current score in a **MsgBox**. This message is to be in the following form: `"11 out of 15"`

3.10 Write code in the **mnuOptions_Level_Click** event sub procedure of **frmCalculator** to carry out the following tasks:

- Show **frmLevel** as a modal dialog with the state of the option buttons set to indicate the current difficulty level.
- Update the **lblDiffLevel** caption.

3.11 Write code in the **mnuOptions_Operator_Click** event sub procedure of **frmCalculator** to carry out the following tasks:

- Show **frmOperator** as a modal dialog with the state of the option buttons set to indicate the current arithmetic operator.
- Update the **lblOperator** caption.

3.12 Write code in the **mnuExit_Click** event sub procedure of **frmCalculator** to unload all forms.

3.13 Write code in the **Form_Load** event sub procedure of **frmLevel** to position the dialog below **frmCalculator** and horizontally centred.

3.14 Write code in the **cmdOK_Click** event sub procedure of **frmLevel** to carry out the following tasks:

- Update the **DifficultyLevel** variable from the state of the option buttons.
- Hide the dialog.

3.15 Write code in the **cmdCancel_Click** event sub procedure of **frmLevel** to hide the dialog without making changes.

3.16 Write code in the **Form_Load** event sub procedure of **frmOperator** to carry out the following tasks:

- Position the dialog below **frmCalculator** and centre it horizontally.
- Use **Line** Method to draw four lines to create the effect of a raised panel containing the four option buttons. The x,y locations of the lines are to be identical to the panel lines on **frmLevel** drawn at design time. Ensure that the **AutoRedraw** property of the form is set to True.

3.17 Write code in the **cmdOK_Click** event sub procedure of **frmOperator** to carry out the following tasks:

- Update the Operator variable from the state of the option buttons.
- Hide the dialog.

3.18 Write code in the **cmdCancel_Click** event sub procedure of **frmOperator** to hide the dialog without making changes.

3.19 Place comments in the program that contain your name and the date. Save your work to a diskette with the name **CALCULTR_1.MAK**. Make sure that your name, the project name and the date are written on the diskette label and hand them to your tutor.

4. MARKING

4.1	Completed within four hours.	[ ]
4.2	Menu system for **frmCalculator** is correct.	( )
4.3	**.bas** module created and saved with the project as **calcultr.bas**	[ ]
4.4	Global variables **Operator** and **DifficultyLevel** declared correctly.	[ ]
4.5	Global constants correctly declared in the **.bas** module.	[ ]

frmCalculator

4.6	Variables correctly declared.	[ ]
4.7	Code in **frmLoad** event sub procedure carries out required tasks.	[ ]
4.8	Code in **mnuCalc_New_Click** event sub procedure carries out required tasks.	[ ]
4.9	Code in **mnuCalc_Score_Click** event sub procedure displays correct message.	( )
4.10	Code in **mnuOptions_Level_Click** event sub proc carries out required tasks.	( )
4.11	Code in **mnuOptions_Operator_Click** event sub proc carries out required tasks.	( )
4.12	Code in **mnuExit_Click** event sub procedure unloads all forms.	( )

frmLevel

4.13	Code in **Form_Load** event sub procedure correctly positions dialogue.	( )
4.14	Code in **cmdOK_Click** event sub procedure carries out required tasks.	( )
4.15	Code in **cmdCancel_Click** event sub procedure correctly hides dialogue.	( )

frmOperator

4.16	Code in **Form_Load** event sub procedure carries out required tasks.	( )
4.17	Code in **cmdOK_Click** event sub procedure carries out required tasks.	( )
4.18	Code in **cmdCancel_Click** event sub procedure correctly hides dialogue.	( )
4.19	Material handed in and disk files are correct.	[ ]
4.20	Program runs correctly.	( )

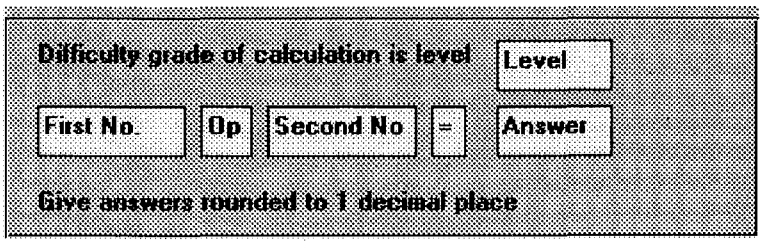
5. ASSIGNMENT COMPLETION

The candidate will have satisfactorily completed this assignment if successful in all items marked with a [ ] and at least eight of the items marked with a ( ).

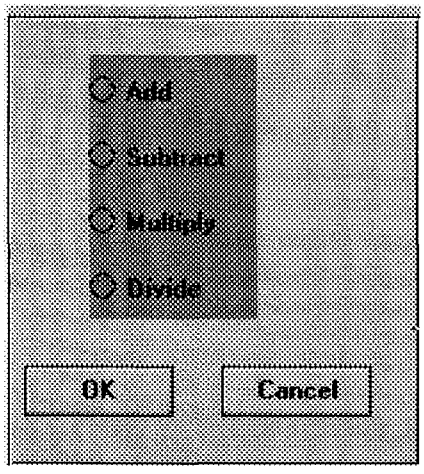
A period of at least seven days must elapse before an unsuccessful candidate may retake this assignment.

6 ASSIGNMENT DOCUMENTATION

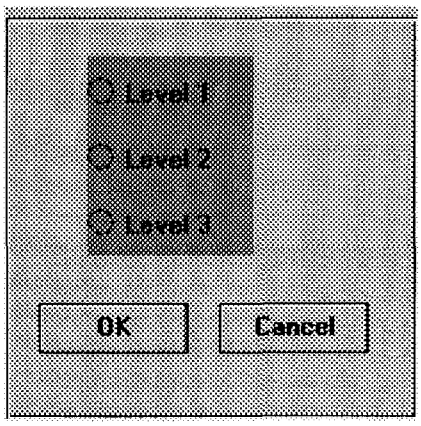
6.1 Images of **frmCalculator**, **frmOperator** and **frmLevel**.



frmCalculator



frmOperator



frmLevel

PRACTICAL ASSIGNMENT: USE OF DRAG AND DROP

1. OBJECTIVE REFERENCES

2.1.1, 2.1.2, 2.2.1, 2.2.2, 4.2.1, 4.2.2, 4.3.1, 4.5.1, 4.9.1, 5.1.1, 5.4.1, 5.5.1, 7.1.2, 7.2.2, 7.2.3, 8.3.3, 9.1.1, 9.1.2, 9.1.3, 9.1.4

2. PREPARATION

2.1	Location of test	The training centre or other venue where supervision and appropriate working conditions will be provided.
2.2	Requirements	Microcomputer or Work station with VGA or SVGA colour monitor running Windows GUI version 3.0 or later version and Visual Basic version 3.0 or later version. User manual for Visual Basic programming language. Copy of para 6.1 A project file **pa03.mak** and the form file **formpa03.frm** The project is to be completely un-coded but controls should have been drawn on the form and properties set as given in para 6.1 A text file named **parts.txt**, either located in the project directory or in a location of which the candidate is advised. (Code is provided in appendix D and electronic files are also available, see note 2 on page Intro/1 of this module)
2.3	Tutor notes	This assignment may be taken over more than one session. Tutors must ensure that all candidate materials are collected at the end of each session and that the work presented represents only that produced by the individual candidate in the time allocated. Network systems may be used, but care must be taken that candidates have no access to their electronic files outside the working sessions. If the machine does not have floppy disk copy facilities the tutor must inform the candidates how they must save their work for marking.

3. CANDIDATE'S INSTRUCTIONS

3.1 The time allowed for this practical assignment is four hours.

You will have been supplied with a project named **pa03.mak** and a sequential text file named **parts.txt** that contains data for the project. If **parts.txt** is not in the project directory you will be advised where to locate it. The project is based on a single form, **formpa03.frm**. The form properties have been set to suit the project and it contains the following controls:

- a Grid control named **grdParts** that uses the file **grid.vbx**
- a Label control named **lblInfo**
- a List Box control named **lstTrans**
- a Command Button named **cmdCommit**
- an Image control named **imgDragIcon** that has its Picture property set to an icon named **drag1pg.ico**
- an Image control named **imgDropIcon** that has its Picture property set to an icon named **drop1pg.ico**

In this assignment you are required to:

- Transfer data from **parts.txt** into the grid control when the form is loaded.
- Use the Drag method to transfer data from the grid to the list box.
- Transfer data from the list box to a sequential text file named **tran.txt** when the Commit button is clicked, and then clear the list box.
- Code and use a procedure to split a string into separate components.
- Update a message in the label control to inform the user as the mouse pointer is moved over certain controls.
- Produce a menu having a single option, to exit the application

The data in **parts.txt** comprises a part number and a description for each of a number of items.

Save the project at regular intervals as you work through the assignment.

3.2 Make the following control property settings:

(i) Set the Visible property of the Image controls to False.
(ii) Set the Enabled property of the Command Button to False.
(iii) Set the form Caption to PA03 (followed by your name and date)

3.3 Produce code that will be executed in response to the **Form_Load** event to carry out the following:

3.3.1 Set the number of fixed rows in the grid to 1.
3.3.2 Set the grid column 1 heading to **Part Number** and its width to 1500; set the column 2 heading to **Description** and its width to 4000. These widths may need subsequent adjustment to suit the file data.
3.3.3 Assign the Picture property of the **imgDragIcon** control to the DragIcon property of the grid control (i.e. the DragIcon property of the grid will become **drag1pg.ico**).
3.3.4 Use the **FreeFile** function to obtain a file number from the operating system and open the **parts.txt** file for Input.
3.3.5 Using the **Line Input #** statement, read the data from the file into the grid, increasing the number of rows in the grid to match the data in the file.
3.3.6 Close the file.
3.3.7 Write error handling code to trap the error that will occur if the **parts.txt** file is not available. Display the error message "Cannot locate file" & ***pathname***.

3.4 Produce code for the **MouseMove** events of the form, the grid, the list box and the command button which will cause the following messages to appear on the Label control, as the mouse pointer is moved over each control:

- Grid . . . . . "Click on an item and drag to list box to add to transaction file"
- List box . . . "Items dragged here are for inclusion in the transaction file"
- Button . . . . "Click to produce transaction file from list items"
- Form . . . . . clear the label caption

3.5 Declare a form level string variable named *DragText.*

3.6 Produce code for the **MouseMove** event of the grid control to carry out the following when the left mouse button is held down (i.e. when the user is dragging):
3.5.1 Begin dragging using the Drag method.
3.5.2 Build a string which contains the part number from the selected grid row, a Tab character and the description from the selected grid row and assign the string to the *DragText* variable.

3.7 In the **DragOver** event of the list box, assign the Picture property of the **imgDropIcon** control to DragIcon property of the grid control (i.e. the DragIcon property of the grid will become **drop1pg.ico**).

3.8 Produce code for the **DragDrop** event of the list box which will carry out the following:
3.8.1 Restore the DragIcon property of the grid control to **drag1pg.ico**
3.8.2 Add the DragText string to the bottom of the **lstTrans** list.
3.8.3 Enable the Commit button.

3.9 Create a form level sub procedure:

Sub Separate(PartLine As String, PartNo As String, PartDesc As String) As Integer.

The sub procedure is to be called with a line of text from the list box assigned to *PartLine* (this will contain a part number, a Tab character and a part description) and empty strings assigned to *PartNo* and *PartDesc*.

The procedure must separate the *PartLine* string into its part number and part description components and assign these to *PartNo* and *PartDesc* respectively, to be returned by the procedure. The Tab character is to be discarded. No assumptions must be made as to the lengths of the part number and description components, i.e. the procedure must be capable of dealing with varying string lengths.

3.10 Produce code for the **Click** event of the command button to carry out the following:

3.10.1 Use the **MsgBox** function to display a message box with OK and Cancel buttons showing the message "Items in list box will be committed to the transaction file". If the response to this message is Cancel then no further action should be taken, otherwise:

3.10.2 Open a file named "**tran.txt**" for output. Set the path for the file to the same directory as that of the **parts.txt** file.

3.10.3 Write the contents of each line of the list box to the file. Each list box line must be split into a part number and a description using the **Separate** sub procedure and then separately written to the file.

3.10.4 Close the **tran.txt** file.

3.10.5 Empty (clear) the list box.

3.11 Produce a menu bar that has one option, **Exit**, which when selected, either by mouse or the **Alt + X** key combination (hot key) causes the termination of the application.

3.12 Demonstrate the operation of your finished program to the test supervisor. The contents of the **tran.txt** file can be verified by loading it into the Notebook utility supplied with Windows.

3.13 Demonstrate the operation of your error trapping code by changing the name of the input data file from **parts.txt** to **perts.txt** in your code, and then executing the program.

3.14 Add suitable descriptive comments to your code to enable someone who is familiar with Visual Basic to understand the operation of the program. Ensure that code is appropriately indented to improve clarity.

3.15 Copy all files to the diskette and place your name on all of the material as well as the diskette label. Hand this material to your tutor (test supervisor).

4. MARKING

4.1	Completed within four hours.	[ ]
4.2	Property values correctly set	( )
4.3	Code for **Form_Load** event:	
	4.3.1 Fixed rows set to 1	( )
	4.3.2 Column headings and widths correct	( )
	4.3.3 DragIcon property correctly set	( )
	4.3.4 Opens file **parts.txt**	[ ]
	4.3.5 Data from file read into grid	[ ]
	4.3.6 Closes file **parts.txt**	[ ]
	4.3.7 Error trapped for 'file not there', gives message and terminates	( )
4.4	Mouse movement over form and controls gives correct messages in label	( )
4.5	Variable correctly declared at form level	( )
4.6	Code for **MouseMove** event:	
	4.6.1 **drag1pg.ico** icon shows when dragging	( )
	4.6.2 String correctly built and assigned to DragText	( )
4.7	**drop1pg.ico** icon shows when mouse pointer over list box	( )
4.8	Code for **DragDrop** event:	
	4.8.1 Grid DragIcon property restored to **drag1pg.ico**	( )
	4.8.2 DragText string dropped into list box correctly	[ ]
	4.8.3 Commit button enabled	( )
4.9	Sub procedure **Separate** operates correctly	[ ]
4.10	Code for command button **Click** event:	
	4.10.1 Correct message box and response to OK and Cancel buttons	( )
	4.10.2 **tran.txt** file opened correctly for output	[ ]
	4.10.3 Lines from list box separated and written correctly to file	[ ]
	4.10.4 **tran.txt** file closed	( )
	4.10.5 List box cleared	( )
4.11	Menu bar with **Exit** correctly implemented	( )
4.12	Program operates. Contents of **tran.txt** file correct	[ ]
4.13	Error trapping demonstrated and operates correctly	[ ]
4.14	Program code appropriately indented and commented	( )
4.15	Materials and disks handed to tutor. Files on disk correct	[ ]

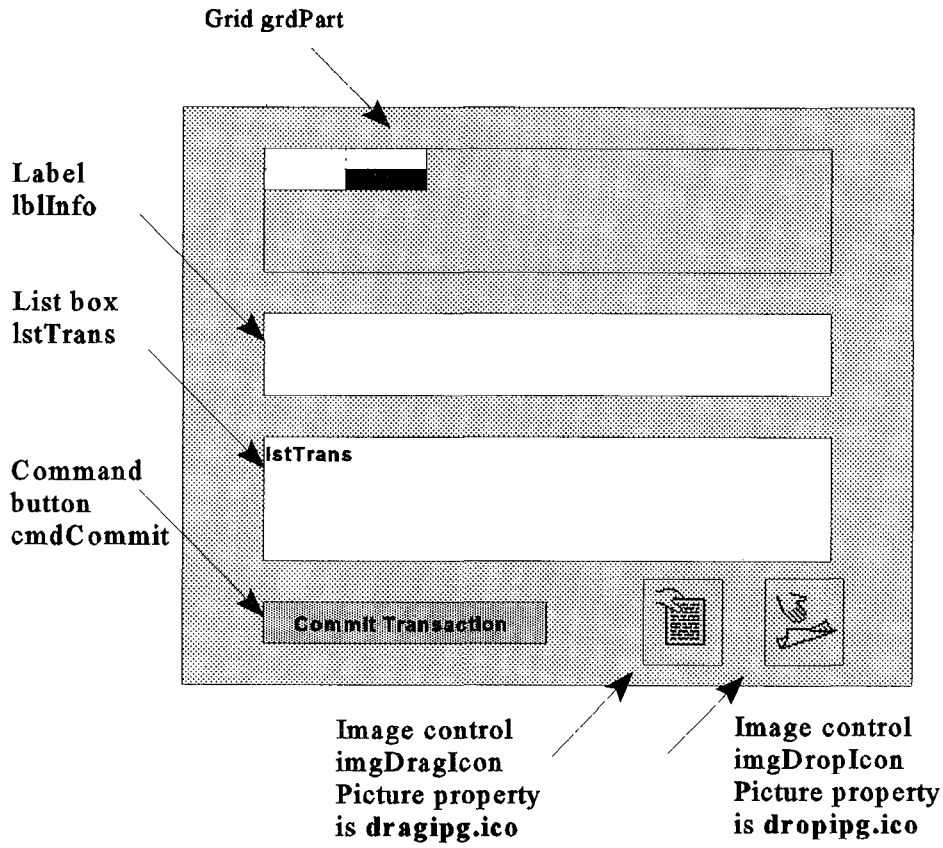
5. ASSIGNMENT COMPLETION

The candidate will have satisfactorily completed this assignment if successful in all items marked with a [ ] and at least twelve of the items marked with a ( ).

A period of at least seven days must elapse before an unsuccessful candidate may retake this assignment.

6 ASSIGNMENT DOCUMENTATION

6.1 Image of design time screen



Note: The Drag/Drop icons are supplied as samples with Visual Basic. They may be found in the \VB\ICONS\DRAGDROP sub-directory.

6.2 **parts.txt**

PARTS.TXT should be created as an Ascii file in Notepad or other editor. It should contain data on about six items of any nature. The data components for each item should be stored on separate lines: the first line holds the part number, the second holds the part description, which need be only three or four words long. A typical example of text for the file is shown below.

```
100
Flat-pack 6-shelf bookcase
200
Adjustable typists chair
300
Four drawer pedestal desk
400
Angle poise lamp
500
Wire basket filing rack
600
Computer storage rack
```

6.3 Control properties that need to be preset

Form

1. *WindowState* = 0
2. *BackColor* to light grey
3. *BorderStyle* = 3

Grid

1. Set *Height* to less than 6 lines so that the vertical scroll bar will show.
2. *ScrollBars* = 2 - Vertical
3. *Cols* = 2
4. *Rows* = 2
5. *DragMode* = Manual
6. *BackColor* to light grey

Images

1. Set *Picture* property of *imgDragIcon* to ***drag1pg.ico**
2. Set *Picture* property of *imgDropIcon* to ***drop1pg.ico**

*These icon files can be found in **\VB\ICONS\DRAGDROP**.

PRACTICAL ASSIGNMENT: WRITE & READ A TEXT FILE AND COUNT THE VOWELS IN A BLOCK OF TEXT

1. OBJECTIVE REFERENCES

2.1.2, 2.2.1, 2.2.2, 4.2.1, 4.2.2, 4.3.1, 4.3.3, 4.3.5, 4.5.1, 4.9.2, 5.2.2, 5.2.3, 5.3.1, 8.1.1, 8.1.2, 8.3.3, 9.1.1, 9.1.2, 9.1.4

2. PREPARATION

2.1	Location of test	The training centre or other venue where supervision and appropriate working conditions will be provided.
2.2	Requirements	Microcomputer or Work station with VGA or SVGA colour monitor running Windows GUI version 3.0 or later version and Visual Basic version 3.0 or later version. User manual for Visual Basic programming language. Copy of para 6.1
2.3	Tutor notes	The tutor must provide the candidate with a copy of the screen image accompanying this PA (**vowels.mak**) to show the layout of the required screen display, para 6.1. (Code is provided in appendix D and electronic files are also available, see note 2 on page Intro/1 of this module) This assignment may be taken over more than one session. Tutors must ensure that all candidate materials are collected at the end of each session and that the work presented represents only that produced by the individual candidate in the time allocated. Network systems may be used, but care must be taken that candidates have no access to their electronic files outside the working sessions. If the machine does not have floppy disk copy facilities the tutor must inform the candidates how they must save their work for marking. The tutor may demonstrate a working version of the program to ensure that candidates understand what is expected of them.

3. CANDIDATE'S INSTRUCTIONS

Part 1

3.1 This (complete) assignment must be completed within 4 hours. You are advised to read all of the candidate's instructions before commencing work.

3.2 You will have been supplied with a partially constructed project called **vowels** based on two forms, **Form1** and **Form2.** Form1 contains a text box (**txt1**), two label controls (**lblLineCount** and **lblVowelCount**) and a command button (**cmdClose**). **Form2** contains a similar set of controls named **txt2**, **lblVowelCount**, **lblStart** and **cmdClose**. Additional labels indicate the purposes of the label and text controls listed above.

View each form in turn and locate it roughly in the centre of the screen .

You are required to code this project to obtain the following functions.

- As text is entered into txt1 all vowels must be converted to upper case (AEIOU) and all other letters must be converted to lower case.
- When the Enter key is pressed the line of text in **txt1** must be saved to a sequential file. The text box must then be cleared ready for the next line.
- A running total of the number of vowels in the saved text must be shown in **lblVowelCount**.
- A running total of the number of lines of text saved to file must be shown in **lblLineCount**.
- When six lines of text have been entered and saved the file must be closed, **Form1** must be hidden and **Form2** shown.
- As **Form2** loads, the text saved previously must be read from the disk file in lines into an array variable, and then displayed in **txt2.**
- When **lblVowelCount** on **Form2** is clicked the number of vowels in the text which has been selected in **txt2** (by the user dragging the mouse) must be displayed in **lblVowelCount.** The location of the first letter of the selected text and the length of the selected text must be displayed in **lblStart**.

The following instructions require certain variables to be declared. In addition you are expected to declare and use other variables and constants as necessary.

3.3 Program the **cmdClose** buttons to stop the project when clicked.

3.4 Set the appropriate property to enable **txt2** to display multiple lines of text.

3.5 Program the **txt1 KeyPress** event to convert vowels to upper-case and all other letters to lower-case. Thus if the user types `apples` or `APPLES`, **txt1** will display `ApplEs`. The following ASCII conversion table may prove helpful: Note that there is a constant difference of 32 between upper and lower case letters.

A 65	B 66	C 67	D 68	E 69	F 70	G 71	H 72	I 73	J 74	K 75	L 76	M 77
N 78	O 79	P 80	Q 81	R 82	S 83	T 84	U 85	V 86	W 87	X 88	Y 89	Z 90
a 97	b 98	c 99	d 100	e 101	f 102	g 103	h 104	I 105	j 106	k 107	l 108	m 109
n 110	o 111	p 112	q 113	r 114	s 115	t 116	u 117	v 118	w 119	x 120	y 121	z 122

3.6 Declare three integer type variables (**Filenumber**, **LineCount** and **VowelCount**) in the general declarations section of **Form1**; the first to hold a file number, the second to hold the running count of lines saved to file and the third to hold the running count of vowels in the saved text.

3.7 Program the **Form1_Load** event sub procedure to perform the following actions.

- Initialise the module variables to zero.
- Initialise **lblVowelCount.Caption** to "vowels = 0"
- Initialise **lblLineCount.Caption** to "lines = 0"
- Obtain a file number from the operating system using the **Freefile** function and open a sequential file for output named **"lines.txt"**

3.8 Add a code module to the project and name it **vowels.bas**

3.9 Add a function (i.e. a function type procedure) called **CountVowels** to **vowels.bas**. Program the function to perform the following actions.

- Accept a string as an argument.
- Return as an integer value the number of vowels in the string.

3.10 Program the **txt1 KeyDown** event sub procedure to perform the following actions.

- If the user presses the **Enter** key then:
  - call the **CountVowels** function to determine the number of vowels in the text in **txt1**;
  - add the vowel count to **VowelCount** and display the value of the variable in **lblVowelCount** in the form "vowels = n" where 'n' represents the vowel count;
  - increment the **LineCount** variable and display the value of the variable in **lblLineCount** in the form "lines = n" where 'n' represents the line count';
  - use the **Write** statement to write the content of **txt1** to **"lines.txt"**, removing any leading or trailing spaces first;
  - clear the text box ready for the next entry.
- When the sixth line is saved to **"lines.txt"** the file must be closed, **Form1** must be hidden and **Form2** must be shown. Thus after six lines are saved the file is closed and activities are transferred to **Form2.**

3.11 Check that the project runs correctly as follows.

- type `apples` ..... displayed as ..... `ApplEs`
- press **Enter** ..... count labels display .. `lines = 1, vowels = 2`
- type `BANANAS` ..... displayed as ..... `bAnAnAs`
- press **Enter** ..... count labels display .. `lines = 2, vowels = 5`
- type `OVER and UNDER` displayed as ..... `OvEr And UndEr`
- press **Enter** ..... count labels display .. `lines = 3, vowels = 10`
- press **Enter**, **Enter**, **Enter** (to bring lines of text saved to 6)
  - **Form1** is hidden;
  - **Form** 2 shows.

3.12 In the **Form2_Load** event sub procedure declare a one dimension array variable with 6 elements named **TxtLine()** to hold the six lines of text saved to file.

3.13 Program **Form2_Load** event sub procedure to perform the following actions.

- Obtain a file number from the operating system using the **Freefile** function and open the file **"lines.txt"** for input.
- Use the **Line Input#** statement to read six lines of text from the file into **TxtLine()**.
- Display the text stored in **TxtLine()** as six separate lines of text in **txt2.**

3.14 Program the **lblVowelCount_Click** event sub procedure to perform the following actions.

- in **lblVowelCount** display "vowels = n" where 'n' represents the number of vowels in the text that has been selected (by dragging the mouse) in **txt2.**
- in **lblStart** display "start = n1, length = n2" where n1 represents the start position of the selected text and n2 represents the length of the selected text.

3.15 Check that the project runs correctly as follows:

- enter the following six lines of text:

```
apples
BANANAS
OVER and UNDER
toast AND jam
eggs, bacon and sausages
Tomatoes are Red
```

- **Form1** is hidden, **Form2** shows and **txt2** displays:

```
"ApplEs"
"bAnAnAs"
"OvEr And UndEr"
"tOAst And jAm"
"Eggs, bAcOn And sAUsAgEs"
"tOmAtOEs ArE rEd"
```

- in **txt2** select text (by dragging the mouse) from the start of UndEr in line 3 to the end of bAcOn in line 5.
- click the **lblVowelCount** box

  lblVowelCount displays ..... vowels = 9

  lblStart displays .... start = 31, length = 37.

3.16 Save your work to a diskette. Ensure that your name is on all materials, including test and rough work material, and give it to your tutor.

4. MARKING

4.1	Completed within four hours.	[ ]
4.2	Project forms positioned roughly at screen centre.	( )
4.3	Close buttons close the project when clicked.	( )
4.4	txt2 displays multiple lines of text.	[ ]
4.5	Vowels converted to upper case, other letters to lower case as entered.	[ ]
4.6	Variables correctly declared at form level.	( )
4.7	**Form1_Load** event sub procedure operates correctly.	[ ]
4.8	**vowels.bas** created.	( )
4.9	Function returns integer value of vowels in the string.	[ ]
4.10	**txt1_KeyDown** event procedure correct up to and including 6th [Enter] press.	[ ]
4.11	Project runs correctly for test given in 3.11.	[ ]
4.12	Array variable correctly declared in **Form2_Load** event sub procedure.	( )
4.13	**Form2_Load** event sub procedure operates correctly.	[ ]
4.14	**lblVowelCount_Click** event sub procedure operates correctly.	[ ]
4.15	Project runs correctly for test given in 3.16.	[ ]
4.16	Materials handed in and disk contains required project files.	[ ]

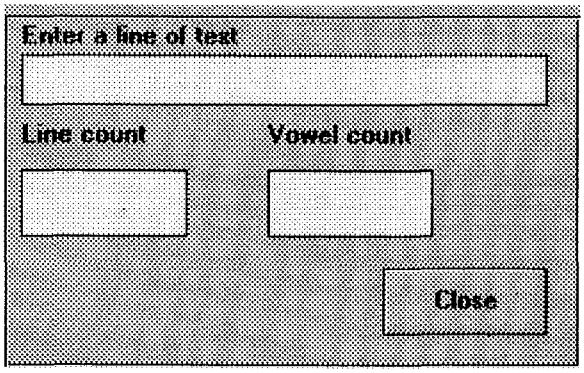
5. ASSIGNMENT COMPLETION

The candidate will have satisfactorily completed this assignment if successful in all items marked with a [ ] and at least 2 of the items marked with a ( ).

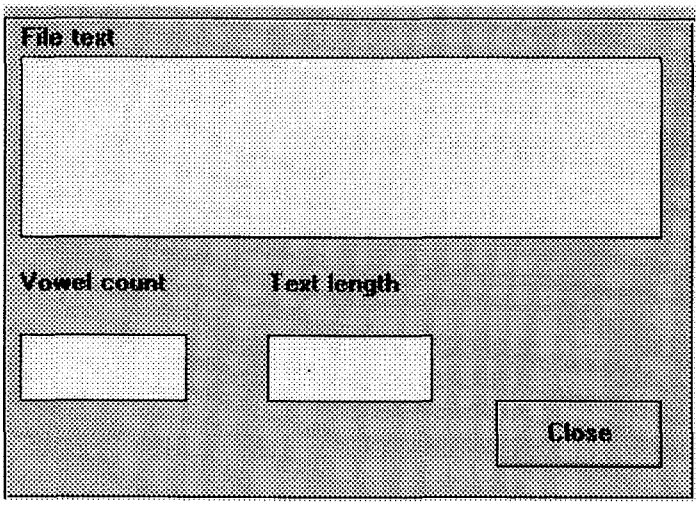
A period of at least seven days must elapse before an unsuccessful candidate may retake this assignment.

6 ASSIGNMENT DOCUMENTATION

6.1 Image of forms; **Form 1** and **Form 2**.



Form 1



Form 2

APPENDIX A

VISUAL DESIGN CONCEPTS

GUIDING PRINCIPLES

Visual Processing Arc

When first viewing an image the eye tends to follow a 'Visual Processing Arc' that runs in a curve from top left to bottom right.

Attractiveness to the eye

The way in which items are presented on screen is important. Because of television and other experiences, the eye is specifically attracted to certain aspects of presentation. For example:

isolated elements are more attractive than groups
colour is more attractive than black and white text
graphics are more attractive than text

Intuitiveness of Interface

A graphical user interface should function intuitively: It should look the way it works and work the way it looks.

WINDOWS PRINCIPLES

General Consistency

Consistency between Windows applications and windows within an application is important, enabling the user to learn a new application more quickly. The following standards provide this consistency of approach

3D effects are used to emphasise functions and give some form of real-world feedback to the user.

Protruding objects can always be pressed or acted upon.

Receding objects are not available, they are static or inactive.

Three dimensional effects are based on an apparent light source which is assumed to be diagonally down from the top left.

White, light grey, dark grey and black are the minimum required to create a clear 3D appearance at VGA resolution.

Black outlines to icons, frames, buttons and controls are used to give an impression of crispness.

Colour

Colours are seen in relationship to other colours around them. The perception of colour depends on hue, saturation and luminance. The following definitions apply:

Hue is the colour defined by the wavelength of its light.

Saturation is the degree to which a colour departs from white and approaches its pure colour. Dull or pale colours are said to have low saturation, vivid colours high saturation.

Luminance (brightness) is a measure of where the colour falls on a scale from light to dark. (White to black)

Colour recommendations

Use colour to show relatedness or grouping. Do not associate colour with meaning, except as a redundant cue.

Avoid using opponent colours together, eg. red and green, yellow and blue, as it is difficult to focus on them.

Avoid large areas of bright colour, which tend to cause after images.

Use subtle colour, and only use it to indicate activity or selection, e.g. active window title bar.

Blue is poor for small text as it is hard to focus on, but it is good as a background colour.

Study the optional Windows colour schemes. They have been carefully devised with colour in mind.

System colours

The basic 16 system system colours in standard VGA mode are as follows:

Colour	RGB values			Colour	RGB values		
	R	**G**	**B**		**R**	**G**	**B**
Yellow	255	255	0	Blue	0	0	255
Dark Yellow	128	128	0	Dark Blue	0	0	128
Red	255	0	0	Cyan	0	255	255
Dark red	128	0	0	Dark Cyan	0	128	128
Magenta	255	0	255	Green	0	255	0
Dark magenta	128	0	128	Dark green	0	128	0
Grey	192	192	192	White	255	255	255
Dark Grey	64	64	64	Black	0	0	0

Extra SVGA colours

Light yellow	255	255	128	Light green	128	255	128
Light blue	128	255	255	Medium grey	128	128	128

Note: In the above table the order of the colour components is Red, Green, Blue. When specifying a colour as a 6-digit hexadecimal value the syntax is **&HBBGGRR&**, i.e. the order of the colour components is reversed. Thus the statement to set the back colour of a form to light yellow, for example, would be: **BackColor=&H80FFFF&** Where Hex 80 represents 129 decimal and Hex FF represents 255 decimal.

Alternatively, using the RGB function, the syntax to set the back colour of a form to Dark Grey would be **BackColor=RGB(0,128,128)**

Fonts

Vary size and weight to indicate hierarchy

Italic and serif fonts tend to be harder to read on screen.

A bold font is necessary if it is to be seen clearly when greyed.

Standard Windows font sizes are:

Menus	10 point, sans serif bold
Dialog boxes	8 point, sans serif bold
Status bars	10 point, sans serif non-bold
Icons	8 point sans serif non-bold

Windows button types

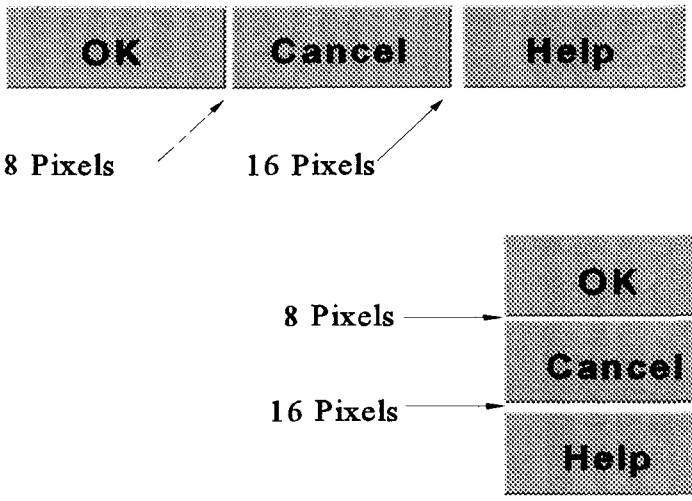
Text buttons have a text caption, e.g. **Cancel**

Graphic buttons have a picture, e.g. for printer. 

Control buttons have a functional symbol, e.g. an arrow 

Text buttons are used in dialog boxes, forms and toolbars. The standard button height for dialog boxes is 22 pixels, the height being determined by the font used.

The recommended horizontal and vertical spacing of a text button is shown below. Less space is used between buttons with a related function than between groups of buttons.



Graphic buttons are used in toolbars and navigation controls. The standard size is 24 pixels wide by 22 pixels high.

APPENDIX B

VISUAL BASIC LEVEL II - LANGUAGE REFERENCE

CONTROLS AND PROPERTIES USED IN THIS MODULE

Controls

Window controls
Control Menu Box Maximise button Minimise button

Grid Control (VBX control)
Common Dialogue Control (VBX control)
Image and Picture box
List box
Menu
Combo Box

Properties

ActiveControl
ActiveForm
AddItem
Autosize
AutoRedraw
Cancel
Checked
Col
ColWidth
DragIcon
Default
DrawWidth
FixedRows
Icon
Index
KeyPreview
ListCount
ListIndex
MousePointer
Picture
Row
Rows
Selected
SelLength
SelStart
SelStartRow
SelText
TabIndex
TabStop
TextWidth
TwipsPerPixelX,Y
Value
Width

EVENTS AND METHODS

Events

Activate	Load
Change	LostFocus
Deactivate	MenuClick
DragDrop	MouseDown
DragOver	MouseMove
Error	MouseUp
GotFocus	Resize
KeyDown	SelChange
KeyPress	Unload

Methods

Circle	Print
Clear	PrintForm
Cls	SetFocus
Drag	Show
Hide	Hide
Line	Zorder

CODING

Statements

Visual Basic statements are normally written one to a line, with no statement terminator. Multiple statements can be written on one line using colons (:) as separators.

Assignment statement

General form *destination=source* to assign a value to a variable or to a property.

Declaring variables and constants

Option Explicit	Dim	Static
Global		

Scope and lifetime of variables and constants

Local	Module	Global

Data types used in this module

Type	Description	Type declaration character
Integer	2-byte	%
Long	4-byte	&
Single	4-byte floating point	!
Double	8-byte floating point	#
String	String of characters	$
Variant	Date, time floating point or string	(none)

Array variables

Single dimension array

Operators used in this module:

Arithmetic

+	-	/
*	( )	

Relational

=	<	>
< =	> =	< >

Logical

AND	OR	NOT

Visual Basic Functions

Err	Int	LoadPicture	Randomise	Rnd	Val

String manipulation

Asc	Chr$	InStr	Left$
Len	LTrim$	Mid$	Right$
Trim$	RTrim$	StrComp	

Control structures

Call	End	Stop

For...Exit For...Next
If...Then...ElseIf...Else...End If
Do...While|Until...Exit Do...Loop
Select Case...Case...Case Is...Case Else...End Select
GoTo.... (Only used for error handling)

Invoking control event procedures from code

Setting the Value property Calling the event procedure

MONITORING & RESPONDING TO USER INPUT

Message and input boxes

MsgBox (Statement and function)
InputBox$

Access letters

Placing an ampersand (**&**) before a letter in a menu or control caption makes that letter the access letter (Alt+X) for the control.

GRAPHICAL METHODS

Screen system, objects & co-ordinates

Screen object properties
Co-ordinate properties:
Screen Form Container

Text

Print

Draw lines and circles

Circle Line Cls

TESTING, DEBUGGING AND ERROR HANDLING

Environment

Immediate pane - Break mode
Watch expressions

Code

OnError...GoTo...Resume...Resume Next...Exit Sub
Err Erl Error Error$

MULTIPLE-FORM APPLICATIONS

Manipulating objects

Show	Hide
Load	Unload
ActiveControl	ActiveForm

FILE HANDLING

Sequential text files

File read

Open	Close#
Input	Input$

File write

Open	Close#
Print #	Write #

Picture files (BMP, WMF)

LoadPicture

ARGUMENTS PASSED TO EVENT PROCEDURES

Keyboard

KeyAscii	KeyCode	Index
Shift		

Mouse

Button	Index	Shift
X, Y		

APPENDIX C

VISUAL BASIC CO-ORDINATE SYSTEM

UNITS OF MEASUREMENT

The default **scale** or unit of measurement is the **'Twip'**. (This is a 'Printed output' measurement.)

1 Twip = 1/1440 inches = 0.0007 inches (Approx)

Printed characters are usually referred to in **'Points'**.

1 Point = 1/72 inches = 0.014 inches (Approx)

Thus the distance from the top of the ascenders to the bottom of the descenders of text printed using 12 point character = 12/72 inches = 0.16 inches (Approx)

20 Twips = 1 Point

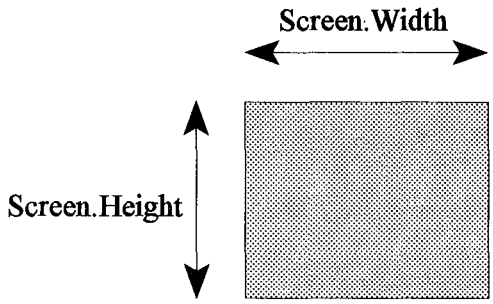
Screen dimensions are usually measured in **'Pixels'**.

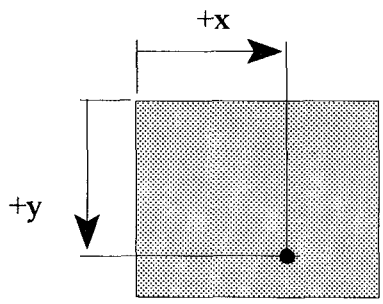
Pixels have no direct relationship to linear measurement in inches. The relationship is dependant upon the resolution of the screen and the mask size used to create the screen.

Common screen resolutions and their (approximate) relation to Twips are:

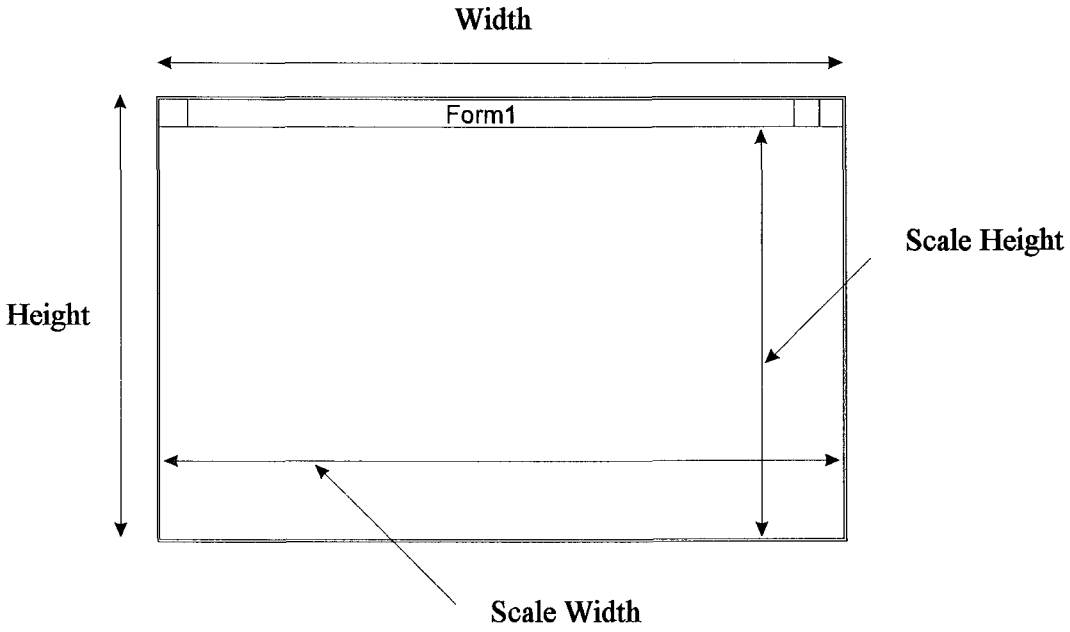
	PIXELS	TWIPS PER PIXEL	SCREEN TWIPS
VGA =	640 Horizontal	15	Screen.Width = 9600
	480 Vertical	15	Screen.Height = 7200
SVGA =	800 Horizontal	15	Screen.Width = 12000
	600 Vertical	15	Screen.Height = 9000

SCREEN OBJECTS





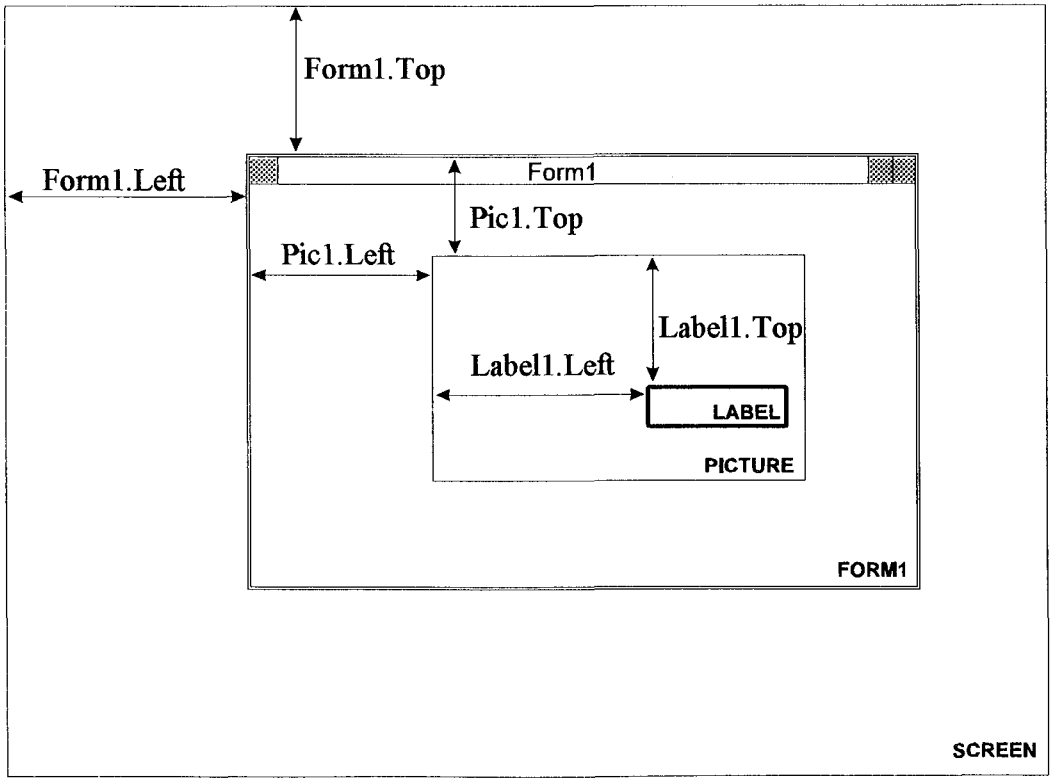
FORM OBJECT



Height and Width are external dimensions including borders and title bar. **ScaleHeight** and **ScaleWidth** are the internal dimensions of the client area. **ScaleHeight** and **ScaleWidth** may be set to custom units in conjunction with the **ScaleMode** property.

CONTROLS WITHIN CONTAINERS

Forms, frames and picture controls can contain other controls. Contained controls move with the container. The **Top** and **Left** properties of a control defines its position relative to its container. The figure below shows a Label control (Label1) contained in a picture control (Pic1) on Form1.



COMPONENT POSITIONS

The positions of each component with respect to one another in the above picture are as follows:

Position of Form1 with respect to screen
= **Form1.Top, Form1.Left**

Position of picture 1 with respect to the screen
= **Form1.Top+Pic1.Top, Form1.Left+Pic1.Left**

Position of label with respect to the screen
= **Form1.Top+Pic1.Top+Label1.Left, Form1.Left+Pic1.Left+Label1.Left**

LOCATION OF MOUSE POINTER

Pressing the left mouse button fires the **MouseDown** event and returns the X, Y location of the mouse pointer in the client area of the control receiving the event. This control then captures the mouse. (i.e. the subsequent MouseUp event will be received by this control). When the **MouseUp** event fires, the X, Y location of the mouse pointer is again returned relative to the control that captured the mouse.

If the mouse pointer is outside the boundary of the control when the button is released then the X and/or Y values will be negative if the mouse is above or to the left of the control or will exceed the Control.Height, Control.Width values when the mouse is below or to the right of the control.

APPENDIX D

CODING FOR PRACTICAL ASSIGNMENTS

Introduction

The four sections of this appendix are printed versions of the form(s), controls, property values and/or code for the programs required in each of the practical assignments. Electronic files of this code are available on diskette, see page Intro/1.

CODE FOR PRACTICAL ASSIGNMENT PA-01

frmDraw CODE

Notes:

The command buttons on **frmDraw** must be created as a control array.
The code in *frmDraw_Load* event should position the command buttons satisfactorily on both VGA and SVGA (800x600) resolution screens, but may require some adjustment.

```
VERSION 2 00
Begin Form frmDraw
  Caption       = "Circle Draw Method"
  ClientHeight  = 4365
  ClientLeft    = 1455
  ClientTop     = 1605
  ClientWidth   = 6465
  Height        = 4800
  Left          = 1380
  LinkTopic     = "Form1"
  ScaleHeight   = 4365
  ScaleWidth    = 6465
  Top           = 1245
  Width         = 6615
  Begin CommandButton cmdTask
    Caption     = "Exit"
    Height      = 408
    Index       = 4
    Left        = 288
    TabIndex    = 4
    Top         = 2304
    Width       = 1596
  End
  Begin CommandButton cmdTask
    Caption     = "Erase"
    Height      = 408
    Index       = 3
    Left        = 288
    TabIndex    = 3
    Top         = 1800
    Width       = 1596
  End
```

```
  Begin CommandButton cmdTask
    Caption     = "Pie Slice"
    Height      = 408
    Index       = 2
    Left        = 288
    TabIndex    = 2
    Top         = 1296
    Width       = 1596
  End
  Begin CommandButton cmdTask
    Caption     = "Ellipse"
    Height      = 408
    Index       = 1
    Left        = 288
    TabIndex    = 1
    Top         = 792
    Width       = 1596
  End
  Begin CommandButton cmdTask
    Caption     = "Circle"
    Height      = 408
    Index       = 0
    Left        = 288
    TabIndex    = 0
    Top         = 288
    Width       = 1596
  End
End
Option Explicit

' CONSTANTS
'----------
' WindowState
Const NORMAL = 0
Const MINIMIZED = 1
Const MAXIMIZED = 2

' Drawing
Const DRAW_CIRCLE = 0      ' value is index of Draw Circle command button
Const DRAW_ELLIPSE = 1     ' value is index of Draw Ellipse command button
Const DRAW_PIE_SLICE = 2   ' value is index of Draw Arc command button
Const CLEAR_CIRCLE = 3     ' value is index of Clear Drawings command button
Const EXIT_PROGRAM = 4     ' value is index of Exit command button

'-------------------------------------------------------------------------------------
' frmDraw
'--------
' This procedure carries out one of the following tasks, depending on which button is clicked·
'
' 1  Show dialog box with correct entry fields for selected drawing type
' 2  Clear existing drawings (erase) from the screen
' 3  Exit program (close application)
'
' Note  Clicking on the form before clicking on one of the drawing buttons will preset
'       the centre point for the drawing in txtCentreX and txtCentreY (see the
'       Form_MouseDown sub procedure).
'-------------------------------------------------------------------------------------
Sub cmdTask_Click (Index As Integer)
Dim N As Integer ' loop counter
```

```
  Select Case Index
    Case DRAW_CIRCLE   ' get circle data
      ' start and end arc and aspect data are not required so hide these fields in the dialog
      For N = 0 To 2  ' hide aspect data labels
        frmDrawDlg!lblAspect(N) Visible = False
      Next
      frmDrawDlg!txtAspectheight Visible = False  ' hide aspect text boxes
      frmDrawDlg!txtAspectWidth Visible = False
      frmDrawDlg!lblStartArc Visible = False     ' hide arc start and end labels and text boxes
      frmDrawDlg!txtStartArc Visible = False
      frmDrawDlg!lblEndArc.Visible = False
      frmDrawDlg!txtEndArc.Visible = False
      frmDrawDlg!lblFillColour Visible = True   ' show fill colour label and textbox
      frmDrawDlg!txtFillColour.Visible = True
      ShowDialog "Draw Circle"   ' show dialog box with appropriate title
      frmDrawDlg!txtCentreX.SetFocus
    Case DRAW_ELLIPSE   ' get ellipse data
      ' start and end arc are not required
      For N = 0 To 2  ' show aspect data labels
        frmDrawDlg!lblAspect(N).Visible = True
      Next
      frmDrawDlg!txtAspectheight.Visible = True   ' show aspect text boxes
      frmDrawDlg!txtAspectWidth Visible = True
      frmDrawDlg!lblStartArc.Visible = False     ' hide arc start and end labels and text boxes
      frmDrawDlg!txtStartArc Visible = False
      frmDrawDlg!lblEndArc Visible = False
      frmDrawDlg!txtEndArc Visible = False
      frmDrawDlg!lblFillColour.Visible = True   ' show fill colour label and text box
      frmDrawDlg!txtFillColour Visible = True
      ShowDialog "Draw Ellipse"   ' show dialog box with appropriate title
      frmDrawDlg!txtCentreX SetFocus
    Case DRAW_PIE_SLICE   ' get arc data
      ' aspect ratio data is not required
      For N = 0 To 2  ' hide aspect data labels
        frmDrawDlg!lblAspect(N) Visible = False
      Next
      frmDrawDlg!txtAspectheight.Visible = False  ' hide aspect text boxes
      frmDrawDlg!txtAspectWidth.Visible = False
      frmDrawDlg!lblStartArc Visible = True     ' show arc start and end labels and text boxes
      frmDrawDlg!txtStartArc Visible = True
      frmDrawDlg!lblEndArc Visible = True
      frmDrawDlg!txtEndArc Visible = True
      frmDrawDlg!lblFillColour Visible = True   ' show fill colour label and text box
      frmDrawDlg!txtFillColour Visible = True
      ShowDialog "Draw Pie Slice"   ' show dialog box with appropriate title
      frmDrawDlg!txtCentreX SetFocus
    Case CLEAR_CIRCLE  ' clear all drawings
      Cls
    Case EXIT_PROGRAM  ' unload all forms
      Unload frmDrawDlg
      Unload frmDraw
  End Select

End Sub

'-------------------------------------------------------------------------------
' frmDraw
'--------
' Prepare the form for drawing operations
'
```

```
' 1  Initialise form properties
' 2  Size and position the drawing command buttons
'    The button sizes and positions are related to the screen dimensions
'
' Note  Clicking on the form before clicking on one of the drawing buttons will preset the
'       centre point for the drawing in txtCentreX and txtCentreY (see the Form_MouseDown
'       sub procedure)
'-------------------------------------------------------------------------------------------
Sub Form_Load ()
Dim N As Integer   ' loop counter
Dim ButtonHeight As String, ButtonWidth As Single   ' button dimensions
Dim ButtonTop As Single  ' vertical location of buttons

   ' maximise the size of frmDraw
   WindowState = MAXIMIZED

   ' set FillStyle property of frmDraw
   FillStyle = 0  ' fill drawings with solid colour

   ' initialise command buttons
   '--------------------------
   ButtonHeight = 1 / 18 * Screen.Height  ' set variable to 1/18 screen height
   ButtonWidth = 1 / 8 * Screen.Width     ' set variable to 1/8 screen width
   ButtonTop =  9 * Screen Height         ' set variable to 9/10 screen height
   For N = DRAW_CIRCLE To EXIT_PROGRAM    ' size command buttons and set vertical positions
      cmdTask(N) Height = ButtonHeight
      cmdTask(N) Width = ButtonWidth
      cmdTask(N) Top = ButtonTop
   Next
   ' separate buttons in group by 27% of button height
   ' separate button groups by 80% of button height
   cmdTask(DRAW_CIRCLE) Left =  14 * Screen Width  ' position leftmost button
   cmdTask(DRAW_ELLIPSE) Left = cmdTask(DRAW_CIRCLE) Left + ButtonWidth + .27 * ButtonHeight
   cmdTask(DRAW_PIE_SLICE) Left = cmdTask(DRAW_ELLIPSE) Left + ButtonWidth +  27 * ButtonHeight
   cmdTask(CLEAR_CIRCLE) Left = cmdTask(DRAW_PIE_SLICE).Left + ButtonWidth +  8 * ButtonHeight
   cmdTask(EXIT_PROGRAM) Left = cmdTask(CLEAR_CIRCLE) Left + ButtonWidth +  8 * ButtonHeight

End Sub

'-----------------------------------------------------------------------
' frmDraw
'--------
' Copy the X, Y position of the mouse pointer to the drawing dialog box
' as the drawing centre for circle, ellipse or pie slice
'-----------------------------------------------------------------------
Sub Form_MouseDown (Button As Integer, Shift As Integer, X As Single, Y As Single)

   frmDrawDlg!txtCentreX Text = CStr(X)  ' convert X value to string and display in txtCentreX
   frmDrawDlg!txtCentreY.Text = CStr(Y)  ' convert Y value to string and display in txtCentreY

End Sub

'-----------------------------------------------------------
' frmDraw
'--------
' Show modal dialog box to get drawing data from user
'-----------------------------------------------------------
Sub ShowDialog (DialogTitle As String)

   frmDrawDlg Caption = DialogTitle  ' the string passed to this procedure is the dialog title
```

```
    frmDrawDlg.Top = Screen.Height - 1 05 * frmDrawDlg.Height   ' position dialog over drawing
                                        ' selection buttons
    frmDrawDlg.Left = (Screen Width - frmDrawDlg.Width) / 2     ' centre dialog horizontally
    frmDrawDlg Show 1  ' show modal dialog

End Sub
```

frmDrawDlg CODE

```
VERSION 2.00
Begin Form frmDrawDlg
  BackColor      = &H00C0C0C0&
  ClientHeight   = 1410
  ClientLeft     = 1320
  ClientTop      = 4155
  ClientWidth    = 7890
  ControlBox     = 0  'False
  Height         = 1845
  Left           = 1245
  LinkTopic      = "Form2"
  MaxButton      = 0  'False
  MinButton      = 0  'False
  ScaleHeight    = 1410
  ScaleWidth     = 7890
  Top            = 3795
  Width          = 8040
  Begin TextBox txtFillColour
    Height       = 300
    Left         = 5640
    TabIndex     = 23
    Top          = 900
    Width        = 900
  End
  Begin TextBox txtLineColour
    Height       = 300
    Left         = 5640
    TabIndex     = 22
    Top          = 360
    Width        = 900
  End
  Begin TextBox txtAspectWidth
    Height       = 300
    Left         = 4380
    TabIndex     = 21
    Top          = 900
    Width        = 900
  End
  Begin TextBox txtAspectheight
    Height       = 300
    Left         = 4380
    TabIndex     = 20
    Top          = 360
    Width        = 900
  End
```

```
Begin TextBox txtEndArc
  Height      = 300
  Left        = 2940
  TabIndex    = 19
  Top         = 900
  Width       = 900
End
Begin TextBox txtStartArc
  Height      = 300
  Left        = 2940
  TabIndex    = 18
  Top         = 360
  Width       = 900
End
Begin TextBox txtDrawWidth
  Height      = 300
  Left        = 1656
  TabIndex    = 17
  Top         = 900
  Width       = 900
End
Begin TextBox txtRadius
  Height      = 300
  Left        = 1656
  TabIndex    = 16
  Top         = 360
  Width       = 900
End
Begin CommandButton cmdDraw
  Caption     = "Draw"
  Height      = 300
  Left        = 6855
  TabIndex    = 15
  Top         = 1005
  Width       = 900
End
Begin CommandButton cmdCancel
  Caption     = "Cancel"
  Height      = 300
  Left        = 6855
  TabIndex    = 14
  Top         = 645
  Width       = 900
End
Begin CommonDialog CMDialog1
  Left        = 7140
  Top         = 150
End
Begin TextBox txtCentreY
  Height      = 300
  Left        = 432
  TabIndex    = 4
  Top         = 900
  Width       = 900
End
```

```
Begin TextBox txtCentreX
  Height      = 300
  Left        = 432
  TabIndex    = 0
  Top         = 360
  Width       = 900
End
Begin Line Line1
  BorderColor  = &H00FFFFFF&
  Index       = 9
  X1          = 6720
  X2          = 6720
  Y1          = 105
  Y2          = 1297
End
Begin Line Line2
  BorderColor  = &H00FFFFFF&
  Index       = 9
  X1          = 5535
  X2          = 6720
  Y1          = 1290
  Y2          = 1290
End
Begin Line Line2
  Index       = 8
  X1          = 5535
  X2          = 6720
  Y1          = 105
  Y2          = 105
End
Begin Line Line2
  BorderColor  = &H00FFFFFF&
  Index       = 7
  X1          = 4050
  X2          = 5418
  Y1          = 1290
  Y2          = 1290
End
Begin Line Line2
  Index       = 6
  X1          = 4050
  X2          = 5418
  Y1          = 105
  Y2          = 105
End
Begin Line Line2
  BorderColor  = &H00FFFFFF&
  Index       = 5
  X1          = 2835
  X2          = 3951
  Y1          = 1290
  Y2          = 1290
End
Begin Line Line2
  Index       = 4
  X1          = 2835
  X2          = 3951
  Y1          = 105
  Y2          = 105
End
```

```
Begin Line Line2
  BorderColor   = &H00FFFFFF&
  Index         = 3
  X1            = 1548
  X2            = 2730
  Y1            = 1300
  Y2            = 1300
End
Begin Line Line2
  Index         = 2
  X1            = 1548
  X2            = 2730
  Y1            = 108
  Y2            = 108
End
Begin Line Line2
  BorderColor   = &H00FFFFFF&
  Index         = 1
  X1            = 108
  X2            = 1440
  Y1            = 1300
  Y2            = 1300
End
Begin Line Line2
  Index         = 0
  X1            = 108
  X2            = 1440
  Y1            = 108
  Y2            = 108
End
Begin Line Line1
  Index         = 8
  X1            = 5535
  X2            = 5535
  Y1            = 105
  Y2            = 1297
End
Begin Line Line1
  BorderColor   = &H00FFFFFF&
  Index         = 7
  X1            = 5430
  X2            = 5430
  Y1            = 105
  Y2            = 1297
End
Begin Line Line1
  Index         = 6
  X1            = 4050
  X2            = 4050
  Y1            = 105
  Y2            = 1297
End
Begin Line Line1
  BorderColor   = &H00FFFFFF&
  Index         = 5
  X1            = 3945
  X2            = 3945
  Y1            = 105
  Y2            = 1297
End
```

```
Begin Line Line1
  Index        = 4
  X1           = 2835
  X2           = 2835
  Y1           = 105
  Y2           = 1297
End
Begin Line Line1
  BorderColor  = &H00FFFFFF&
  Index        = 3
  X1           = 2730
  X2           = 2730
  Y1           = 120
  Y2           = 1312
End
Begin Line Line1
  Index        = 2
  X1           = 1548
  X2           = 1548
  Y1           = 108
  Y2           = 1300
End
Begin Line Line1
  Index        = 1
  X1           = 108
  X2           = 108
  Y1           = 108
  Y2           = 1300
End
Begin Line Line1
  BorderColor  = &H00FFFFFF&
  Index        = 0
  X1           = 1440
  X2           = 1440
  Y1           = 108
  Y2           = 1300
End
Begin Label lblFillColour
  AutoSize     = -1 'True
  BackColor    = &H00C0C0C0&
  Caption      = "Fill Colour"
  Height       = 195
  Left         = 5640
  TabIndex     = 13
  Top          = 690
  Width        = 855
End
Begin Label lblLineColour
  AutoSize     = -1 'True
  BackColor    = &H00C0C0C0&
  Caption      = "Line Colour"
  Height       = 195
  Left         = 5640
  TabIndex     = 12
  Top          = 150
  Width        = 945
End
```

```
Begin Label lblAspect
  AutoSize      = -1 'True
  BackColor     = &H00C0C0C0&
  Caption       = "W"
  Height        = 195
  Index         = 2
  Left          = 4125
  TabIndex      = 11
  Top           = 975
  Width         = 180
End
Begin Label lblAspect
  AutoSize      = -1 'True
  BackColor     = &H00C0C0C0&
  Caption       = "H"
  Height        = 195
  Index         = 1
  Left          = 4170
  TabIndex      = 10
  Top           = 435
  Width         = 150
End
Begin Label lblAspect
  AutoSize      = -1 'True
  BackColor     = &H00C0C0C0&
  Caption       = "Ellipse aspect"
  Height        = 195
  Index         = 0
  Left          = 4170
  TabIndex      = 9
  Top           = 150
  Width         = 1200
End
Begin Label lblEndArc
  AutoSize      = -1 'True
  BackColor     = &H00C0C0C0&
  Caption       = "End Arc"
  Height        = 195
  Left          = 2940
  TabIndex      = 8
  Top           = 690
  Width         = 660
End
Begin Label lblStartArc
  AutoSize      = -1 'True
  BackColor     = &H00C0C0C0&
  Caption       = "Start Arc"
  Height        = 195
  Left          = 2940
  TabIndex      = 7
  Top           = 150
  Width         = 720
End
```

```
  Begin Label Label5
    AutoSize      =  -1  'True
    BackColor     =  &H00C0C0C0&
    Caption       =  "Draw Width"
    Height        =  195
    Left          =  1650
    TabIndex      =  6
    Top           =  690
    Width         =  1005
  End
  Begin Label Label4
    AutoSize      =  -1  'True
    BackColor     =  &H00C0C0C0&
    Caption       =  "Radius"
    Height        =  192
    Left          =  1656
    TabIndex      =  5
    Top           =  144
    Width         =  600
  End
  Begin Label Label3
    AutoSize      =  -1  'True
    BackColor     =  &H00C0C0C0&
    Caption       =  "Y"
    Height        =  192
    Left          =  216
    TabIndex      =  3
    Top           =  936
    Width         =  132
  End
  Begin Label Label2
    AutoSize      =  -1  'True
    BackColor     =  &H00C0C0C0&
    Caption       =  "X"
    Height        =  192
    Left          =  216
    TabIndex      =  2
    Top           =  432
    Width         =  120
  End
  Begin Label Label1
    AutoSize      =  -1  'True
    BackColor     =  &H00C0C0C0&
    Caption       =  "Centre"
    Height        =  192
    Left          =  432
    TabIndex      =  1
    Top           =  144
    Width         =  552
  End
End
Option Explicit
' CONSTANTS
'-----------------------------------------------------------------
' Action property value for Color Dialog (Common Dialog box control)
Const DLG_COLOR = 3

' Approximate value for Pi
Const PI = 3 142
```

```
' Message box type
Const MB_ICONSTOP = 16


'----------------------------
' frmDrawDlg
'-----------
' Hide the drawing dialog box
'----------------------------
Sub cmdCancel_Click ()

   frmDrawDlg Hide

End Sub

'-------------------------------------------------------------------------
' frmDrawDlg
'-----------
' This procedure copies the user's drawing data into local variables from the
' dialog box data fields, validates the entries and draws the chosen shape.
'
' 1  If a data entry is invalid a message box is shown and the user is
'    returned to the data entry dialog
'
' 2  When all data values are valid the user's chosen shape is drawn
'-------------------------------------------------------------------------
Sub cmdDraw_Click ()
' local variables to hold drawing data
Dim DialogTitle As String
Dim CentreX As Long, CentreY As Long
Dim Radius As Long
Dim LineWidth As Integer
Dim LineColor As Long, SolidColor As Long
Dim AspectH As Single, AspectW As Single    ' AspectH/AspectW is used to set
                                            ' the aspect ratio of the ellipse
Dim StartArc As Single, EndArc As Single
' Message box variables
Dim bDataInvalid As Integer ' set True  when invalid data is encountered, False otherwise
Dim Msg As String           ' message string shown in invalid data message box


   ' clear any existing drawing
   frmDraw Cls

   ' get drawing data from the dialog box data entry fields
   CentreX = Val(txtCentreX Text)  ' text entries must be converted to numeric type
   CentreY = Val(txtCentreY Text)
   Radius = Val(txtRadius Text)
   LineWidth = Val(txtDrawWidth.Text)
   LineColor = Val(txtLineColour.Text)
   ' get remaining data values only if data text box is visible
   If txtAspectHeight Visible Then AspectH = Val(txtAspectHeight Text)
   If txtAspectWidth Visible Then AspectW = Val(txtAspectWidth.Text)
   If txtStartArc.Visible Then StartArc = Val(txtStartArc Text)
   If txtEndArc Visible Then EndArc = Val(txtEndArc.Text)
   If txtFillColour Visible Then SolidColor = Val(txtFillColour Text)

   ' Validate entries to ensure no out of range values
   ' Set flag if a data entry is invalid
   '-------------------------------------------------
```

```
If CentreX < 0 Or CentreX > Screen.Width Then bDataInvalid = True
If CentreY < 0 Or CentreY > Screen Height Then bDataInvalid = True
If Radius < 0 Or Radius > Screen Height / 2 Then bDataInvalid = True
If LineWidth < 1 Or LineWidth > 300 Then bDataInvalid = True
If LineColor < 0 Or LineColor > &HFFFFFF Then bDataInvalid = True
If SolidColor < 0 Or SolidColor > &HFFFFFF Then bDataInvalid = True
If txtAspectHeight Visible And (AspectH < 1 Or AspectH > 20) Then bDataInvalid = True
If txtAspectWidth Visible And (AspectW < 1 Or AspectW > 20) Then bDataInvalid = True
If txtStartArc Visible And (StartArc < 0 Or StartArc > 2 * PI) Then bDataInvalid = True
If txtEndArc Visible And (EndArc < 0 Or EndArc > 2 * PI) Then bDataInvalid = True
If bDataInvalid Then  ' show message box if invalid data flag is set
   Msg = "One or more entries is invalid or out of range." & Chr(10) & Chr(10)
   Msg = Msg & "* All entries must be positive " & Chr(10)
   Msg = Msg & "* Centre must be on screen." & Chr(10)
   Msg = Msg & "* Radius cannot exceed half the screen height " & Chr(10)
   Msg = Msg & "* Line width value must be from 1 to 300 " & Chr(10)
   Msg = Msg & "* Aspect ratio values must be from 1 to +2*Pi." & Chr(10)
   Msg = Msg & "* Start and End of arc must be from 0 to +2*Pi."
   MsgBox Msg, MB_ICONSTOP, "Data error"
   bDataInvalid = False  ' reset error flag
   Exit Sub ' allow user to change entry
End If

' set drawing parameters for form
'--------------------------------
frmDraw.DrawWidth = LineWidth
frmDraw FillColor = SolidColor

' draw the selected shape
'------------------------
DialogTitle = Caption   ' get the dialog title to determine user's chosen shape
frmDrawDlg Hide         ' done with dialog, it can be hidden
Select Case DialogTitle ' draw the chosen shape using the Circle method
Case "Draw Circle"
   ' draw circle
   frmDraw Circle (CentreX, CentreY), Radius, LineColor
Case "Draw Elipse"
   ' draw ellipse
   frmDraw Circle (CentreX, CentreY), Radius, LineColor, , , (AspectH / AspectW)
Case "Draw Pie Slice"
   '////////////////////////////////////////////////////////////////////////
   ' Notes
   '     1  The following lines adjust the values of StartArc and EndArc
   '        to ensure that radius lines are drawn to complete the pie slice
   '     2  Radius lines are only drawn if both values are negative
   '     3  A zero value must be converted to a very small negative value
   '        or the corresponding radius line will not be drawn and the slice
   '        will not be filled with colour
   '////////////////////////////////////////////////////////////////////////
   If StartArc = 0 Then
   StartArc = - 001  ' set to a very small negative value
   Else
   StartArc = -1 * StartArc ' set to equivalent negative value
   End If
   If EndArc = 0 Then
   EndArc = - 001   ' set to a very small negative value
   Else
   EndArc = -1 * EndArc ' set to equivalent negative value
   End If
   ' draw the pie slice
```

```
    frmDraw Circle (CentreX, CentreY), Radius, LineColor, StartArc, EndArc
  End Select

'////////////////////////////////////////////////////////////////////////////
' Note
' The syntax used in this procedure for the circle method is shown below based on the
' variables defined in this procedure
' Arguments in the middle of the syntax that are not required may be omitted, but the
' argument's comma must be included before the next argument (see the syntax for
' drawing an ellipse above for an example)
' Trailing arguments may be omitted  in which case the commas must be omitted also
' (see the syntax for drawing a circle above for an example)
'
' Circle (CentreX, CentreY), Radius, LineColor, StartArc, EndArc, (AspectH / AspectW)
'////////////////////////////////////////////////////////////////////////////

End Sub

'---------------------------------------------------------------------------
' Show the Common Dialog Color box and obtain a colour vaLue from the user
'---------------------------------------------------------------------------
Function GetColor () As Long

  ' generate error if user cancels dialog
  CMDialog1 CancelError = True

  On Error GoTo CancelError

  ' show common dialog colour box
  CMDialog1 Action = DLG_COLOR
  ' get the colour value from the dialog Color property
  ' and return it via the funtion name
  GetColor = CMDialog1.Color

  Exit Function

CancelError

  ' user has cancelled the dialog - return -1 from the function to indicate this
  GetColor = -1

  Exit Function

End Function

'---------------------------------------------------------------------
' frmDrawDlg
'-----------
' Call the GetColor function and get a colour selection from the user.
' The function procedure name is used to return the colour value.
'---------------------------------------------------------------------
Sub txtFillColour_DblClick ()
Dim ColorValue As Long

  ColorValue = GetColor()  ' get a colour value from the user

  If ColorValue >= 0 Then  ' a colour value has been returned
    txtFillColour Text = CStr(ColorValue)
  End If
```

```
End Sub

'-------------------------------------------------------------------
' frmDrawDlg
'-----------
' Call the GetColor function and get a colour selection from the user.
' The function procedure name is used to return the colour value.
'-------------------------------------------------------------------
Sub txtLineColour_DblClick ()
Dim ColorValue As Long

   ColorValue = GetColor() ' get a colour value from the user

   If ColorValue >= 0 Then ' a colour value has been returned
      txtLineColour Text = CStr(ColorValue)
   End If

End Sub
```

CODE FOR PRACTICAL ASSIGNMENT PA-02

PA02

Calculator Form
Level Form
Operator Form

```
VERSION 2.00
Begin Form frmCalculator
  BackColor     = &H00E0FFFF&
  Caption       = "Calculator"
  ClientHeight  = 1488
  ClientLeft    = 1488
  ClientTop     = 1296
  ClientWidth   = 5628
  Height        = 1872
  Left          = 1440
  LinkTopic     = "Form1"
  ScaleHeight   = 1488
  ScaleWidth    = 5628
  Top           = 960
  Width         = 5724
  Begin Label lblDivideMsg
    BackStyle     = 0 'Transparent
    Caption       = "Give answers rounded to 1 decimal place accuracy"
    Height        = 192
    Left          = 540
    TabIndex      = 7
    Top           = 1116
    Visible       = 0 'False
    Width         = 4380
  End
```

```
Begin Label lblDiffLevel
  AutoSize      = -1 'True
  BackColor     = &H00E0FFFF&
  Caption       = "Level"
  Height        = 192
  Left          = 3636
  TabIndex      = 6
  Top           = 144
  Width         = 468
End
Begin Label Label1
  BackColor     = &H00E0FFFF&
  Caption       = "Difficulty grade of calculation is level"
  Height        = 192
  Left          = 540
  TabIndex      = 5
  Top           = 144
  Width         = 3096
End
Begin Label lblAnswer
  Alignment     = 2 'Center
  BorderStyle   = 1 'Fixed Single
  Caption       = "Answer"
  FontBold      = -1 'True
  FontItalic    = 0 'False
  FontName      = "MS Sans Serif"
  FontSize      = 9 6
  FontStrikethru = 0 'False
  FontUnderline = 0 'False
  Height        = 300
  Left          = 4068
  TabIndex      = 4
  Top           = 540
  Width         = 1104
End
Begin Label lblEqualSign
  Alignment     = 2 'Center
  BorderStyle   = 1 'Fixed Single
  Caption       = "="
  FontBold      = -1 'True
  FontItalic    = 0 'False
  FontName      = "MS Sans Serif"
  FontSize      = 9 6
  FontStrikethru = 0 'False
  FontUnderline = 0 'False
  Height        = 300
  Left          = 3636
  TabIndex      = 3
  Top           = 540
  Width         = 312
End
```

```
  Begin Label lblSecondNo
    Alignment      = 2 'Center
    BorderStyle    = 1 'Fixed Single
    Caption        = "Second No"
    FontBold       = -1 'True
    FontItalic     = 0 'False
    FontName       = "MS Sans Serif"
    FontSize       = 9.6
    FontStrikethru = 0 'False
    FontUnderline  = 0 'False
    Height         = 300
    Left           = 2232
    TabIndex       = 2
    Top            = 540
    Width          = 1308
  End
  Begin Label lblOperator
    Alignment      = 2 'Center
    BorderStyle    = 1 'Fixed Single
    Caption        = "Op"
    FontBold       = -1 'True
    FontItalic     = 0 'False
    FontName       = "MS Sans Serif"
    FontSize       = 9.6
    FontStrikethru = 0 'False
    FontUnderline  = 0 'False
    Height         = 300
    Left           = 1764
    TabIndex       = 1
    Top            = 540
    Width          = 348
  End
  Begin Label lblFirstNo
    Alignment      = 2 'Center
    BorderStyle    = 1 'Fixed Single
    Caption        = "First No"
    FontBold       = -1 'True
    FontItalic     = 0 'False
    FontName       = "MS Sans Serif"
    FontSize       = 9 6
    FontStrikethru = 0 'False
    FontUnderline  = 0 'False
    Height         = 300
    Left           = 540
    TabIndex       = 0
    Top            = 540
    Width          = 1104
  End
End
Option Explicit


VERSION 2 00
Begin Form frmLevel
  BackColor      = &H00C0C0C0&
  Caption        = "Level of Difficulty"
  ClientHeight   = 2052
  ClientLeft     = 1224
  ClientTop      = 1548
  ClientWidth    = 2280
```

```
Height      = 2436
Left        = 1176
LinkTopic   = "Form3"
ScaleHeight = 2052
ScaleWidth  = 2280
Top         = 1212
Width       = 2376
Begin CommandButton cmdCancel
  Cancel      = -1 'True
  Caption     = "Cancel"
  Height      = 300
  Left        = 1224
  TabIndex    = 4
  Top         = 1656
  Width       = 900
End
Begin CommandButton cmdOK
  Caption     = "OK"
  Default     = -1 'True
  Height      = 300
  Left        = 180
  TabIndex    = 3
  Top         = 1656
  Width       = 900
End
Begin OptionButton optLevel
  BackColor   = &H00C0C0C0&
  Caption     = "Level 3"
  Height      = 300
  Index       = 2
  Left        = 612
  TabIndex    = 2
  Top         = 1008
  Width       = 1100
End
Begin OptionButton optLevel
  BackColor   = &H00C0C0C0&
  Caption     = "Level 2"
  Height      = 300
  Index       = 1
  Left        = 612
  TabIndex    = 1
  Top         = 612
  Width       = 1100
End
Begin OptionButton optLevel
  BackColor   = &H00C0C0C0&
  Caption     = "Level 1"
  Height      = 300
  Index       = 0
  Left        = 612
  TabIndex    = 0
  Top         = 216
  Value       = -1 'True
  Width       = 1100
End
```

```
  Begin Line Line1
    BorderColor    =  &H00000000&
    Index          =  3
    X1             =  2000
    X2             =  2000
    Y1             =  144
    Y2             =  1476
  End
  Begin Line Line1
    BorderColor    =  &H00FFFFFF&
    Index          =  2
    X1             =  250
    X2             =  250
    Y1             =  144
    Y2             =  1476
  End
  Begin Line Line1
    BorderColor    =  &H00000000&
    Index          =  1
    X1             =  250
    X2             =  2000
    Y1             =  1476
    Y2             =  1476
  End
  Begin Line Line1
    BorderColor    =  &H00FFFFFF&
    Index          =  0
    X1             =  250
    X2             =  2000
    Y1             =  144
    Y2             =  144
  End
End
Option Explicit

VERSION 2 00
Begin Form frmOperator
  AutoRedraw     =  -1  'True
  BackColor      =  &H00C0C0C0&
  Caption        =  "Select Operator "
  ClientHeight   =  2052
  ClientLeft     =  3984
  ClientTop      =  1548
  ClientWidth    =  2280
  Height         =  2436
  Left           =  3936
  LinkTopic      =  "Form2"
  ScaleHeight    =  2052
  ScaleWidth     =  2280
  Top            =  1212
  Width          =  2376
  Begin CommandButton cmdCancel
    Cancel         =  -1  'True
    Caption        =  "Cancel"
    Height         =  300
    Left           =  1224
    TabIndex       =  5
    Top            =  1656
    Width          =  900
  End
```

```
  Begin CommandButton cmdOK
    Caption      = "OK"
    Default      = -1 'True
    Height       = 300
    Left         = 180
    TabIndex     = 4
    Top          = 1656
    Width        = 900
  End
  Begin OptionButton optOperator
    BackColor    = &H00C0C0C0&
    Caption      = "Divide"
    Height       = 300
    Index        = 3
    Left         = 648
    TabIndex     = 3
    Top          = 1116
    Width        = 1000
  End
  Begin OptionButton optOperator
    BackColor    = &H00C0C0C0&
    Caption      = "Multiply"
    Height       = 300
    Index        = 2
    Left         = 648
    TabIndex     = 2
    Top          = 792
    Width        = 1000
  End
  Begin OptionButton optOperator
    BackColor    = &H00C0C0C0&
    Caption      = "Subtract"
    Height       = 300
    Index        = 1
    Left         = 648
    TabIndex     = 1
    Top          = 468
    Width        = 1000
  End
  Begin OptionButton optOperator
    BackColor    = &H00C0C0C0&
    Caption      = "Add"
    Height       = 300
    Index        = 0
    Left         = 648
    TabIndex     = 0
    Top          = 180
    Value        = -1 'True
    Width        = 1000
  End
End
Option Explicit
```

CODE FOR PRACTICAL ASSIGNMENT PA-03

PA03

```
VERSION 2 00
Begin Form frmPA03
  BackColor     = &H00C0C0C0&
  BorderStyle   = 3 'Fixed Double
  ClientHeight  = 5664
  ClientLeft    = 2052
  ClientTop     = 2184
  ClientWidth   = 6792
  Height        = 6336
  Icon          = 0
  Left          = 2004
  LinkTopic     = "Form2"
  ScaleHeight   = 5664
  ScaleWidth    = 6792
  Top           = 1560
  Width         = 6888
  Begin CommandButton cmdCommit
    Caption     = "Commit Transaction"
    Height      = 444
    Left        = 648
    TabIndex    = 3
    Top         = 4608
    Width       = 2100
  End
  Begin ListBox lstTrans
    Columns     = 1
    Height      = 1560
    Left        = 648
    TabIndex    = 2
    Top         = 2556
    Width       = 5448
  End
  Begin Grid grdParts
    FixedCols   = 0
    FixedRows   = 0
    Height      = 1092
    Left        = 648
    ScrollBars  = 2 'Vertical
    TabIndex    = 0
    Top         = 180
    Width       = 1992
  End
  Begin Image imgDropIcon
    Height      = 384
    Left        = 5652
    Picture     = FORMPA03.FRX.0000
    Top         = 4716
    Width       = 384
  End
```

```
  Begin Image imgDragIcon
    Height      = 384
    Left        = 5112
    Picture     = FORMPA03 FRX:0302
    Top         = 4680
    Width       = 384
  End
  Begin Label lblInfo
    BorderStyle   = 1 'Fixed Single
    Height      = 588
    Left        = 648
    TabIndex    = 1
    Top         = 1728
    Width       = 5448
  End
  Begin Menu mnuExit
    Caption     = "E&xit"
  End
End
Option Explicit
```

CODE FOR PRACTICAL ASSIGNMENT PA-04

PA04

Form Vowels (Form1)
Form 2

```
VERSION 2.00
Begin Form Form1
  BackColor     = &H00C0C0C0&
  Caption       = "Form1"
  ClientHeight  = 2340
  ClientLeft    = 1956
  ClientTop     = 1596
  ClientWidth   = 5040
  ControlBox    = 0 'False
  Height        = 2724
  Left          = 1908
  LinkTopic     = "Form1"
  MaxButton     = 0 'False
  MinButton     = 0 'False
  ScaleHeight   = 2340
  ScaleWidth    = 5040
  Top           = 1260
  Width         = 5136
  Begin TextBox txt1
    Height      = 372
    Left        = 312
    TabIndex    = 0
    Top         = 360
    Width       = 4452
  End
```

```
  Begin CommandButton cmdClose
    Caption       = "Close"
    Height        = 372
    Left          = 3780
    TabIndex      = 1
    Top           = 1800
    Width         = 972
  End
  Begin Label Label3
    BackColor     = &H00C0C0C0&
    BackStyle     = 0 'Transparent
    Caption       = "Vowel count"
    Height        = 192
    Left          = 2196
    TabIndex      = 6
    Top           = 828
    Width         = 1260
  End
  Begin Label Label2
    BackColor     = &H00C0C0C0&
    BackStyle     = 0 'Transparent
    Caption       = "Line count"
    Height        = 192
    Left          = 324
    TabIndex      = 5
    Top           = 828
    Width         = 1068
  End
  Begin Label Label1
    BackColor     = &H00C0C0C0&
    BackStyle     = 0 'Transparent
    Caption       = "Enter a line of text"
    Height        = 192
    Left          = 324
    TabIndex      = 4
    Top           = 72
    Width         = 1704
  End
  Begin Label lblLineCount
    BorderStyle   = 1 'Fixed Single
    Height        = 372
    Left          = 324
    TabIndex      = 3
    Top           = 1116
    Width         = 1692
  End
  Begin Label lblVowelCount
    BorderStyle   = 1 'Fixed Single
    Height        = 372
    Left          = 2196
    TabIndex      = 2
    Top           = 1116
    Width         = 1692
  End
End
Option Explicit
```

```
VERSION 2 00
Begin Form Form2
  BackColor      = &H00C0C0C0&
  Caption        = "Form2"
  ClientHeight   = 3792
  ClientLeft     = 1992
  ClientTop      = 4428
  ClientWidth    = 5148
  ControlBox     = 0  'False
  Height         = 4176
  Left           = 1944
  LinkTopic      = "Form2"
  MaxButton      = 0  'False
  MinButton      = 0  'False
  ScaleHeight    = 3792
  ScaleWidth     = 5148
  Top            = 4092
  Width          = 5244
  Begin CommandButton cmdClose
    Caption      = "Close"
    Height       = 372
    Left         = 3996
    TabIndex     = 3
    Top          = 3240
    Width        = 876
  End
  Begin TextBox txt2
    Height       = 1692
    Left         = 360
    MultiLine    = -1  'True
    TabIndex     = 0
    Top          = 468
    Width        = 4452
  End
  Begin Label Label3
    BackStyle    = 0  'Transparent
    Caption      = "Text length"
    Height       = 192
    Left         = 2376
    TabIndex     = 6
    Top          = 2304
    Width        = 1104
  End
  Begin Label Label2
    BackStyle    = 0  'Transparent
    Caption      = "Vowel count"
    Height       = 192
    Left         = 360
    TabIndex     = 5
    Top          = 2304
    Width        = 1188
  End
  Begin Label Label1
    BackStyle    = 0  'Transparent
    Caption      = "File text"
    Height       = 192
    Left         = 360
    TabIndex     = 4
    Top          = 144
    Width        = 912
```

```
  End
  Begin Label lblStart
    BorderStyle    =  1  'Fixed Single
    Height         =  372
    Left           =  2376
    TabIndex       =  2
    Top            =  2592
    Width          =  1932
  End
  Begin Label lblVowelCount
    BorderStyle    =  1  'Fixed Single
    Height         =  372
    Left           =  324
    TabIndex       =  1
    Top            =  2592
    Width          =  1692
  End
End
```

Published by City & Guilds
1 Giltspur Street, London EC1A 9DD

IT-22-7261