# BHANDARKARS' ARTS AND SCIENCE COLLEGE KUNDAPURA
## COMPUTER SCIENCE DEPARTMENT
## II – BCA DATA STRUCTURE (2 MARKS QUESTION AND ANSWER)

**1. Mention the four types of applications can be developed in .Net framework.**

Windows Application
Web Application
Console Application
Mobile applications

**2. What is CLR? How it functions?**

At the base of the .NET framework is the Common Language Runtime (CLR). The CLR is the module that actually runs your VB .NET applications. When you create a VB .NET application, what really happens is that your code is *compiled* into the CLR's *Intermediate Language* (named MSIL, or IL for short), much like bytecodes in Java. When you run the application, that IL code is translated into the binary code your computer can understand by some special *compilers* built into the CLR. Compilers translate your code into something that your machine's hardware, or other software, can deal with directly. In this way, Microsoft can one day create a CLR for operating systems other than Windows, and your VB .NET applications, compiled into IL, will run on them.

**3. What is the purpose of namespaces? Mention the use of any 2 namespaces in VB .Net.**

The .NET Framework class library is the second major part of the .NET framework. The class library holds an immense amount of prewritten code that all the applications you create with Visual Basic, Visual C++, C#, and other Visual Studio languages build on.

- ***System.Data*-**Includes classes that make up ADO.NET. ADO.NET lets you build data-handling components that manage data from multiple distributed data sources.
- ***System.Data.SqlClient*-**Includes classes that support the SQL Server .NET data provider.
- ***System.IO*-**Includes types that support synchronous and asynchronous reading from and writing to both data streams and files.
- ***System.Security*-**Includes classes that support the structure of the common language runtime security system.

**4. What are solutions and projects?**

Each project held the code and data for an application, ActiveX control, or whatever else you wanted to build. If you wanted to combine projects together, you created a *project group*. In VB .NET, however, project groups have become far more integral to the development process, and now they're called *solutions*.

when you create a new project in VB .NET, Visual Basic will create a new solution first, and then add a project to that solution.

**5. Mention any four File extensions used in VB .Net.**

*.xml*-An XML document file.

.*htm*-An HTML document.
.*txt*-A text file.
.*aspx*-A Web form.
.*vbs*-A VBScript file.

**6. What is VBIDE? Mention any two components of VBIDE.**

The **VBIDE** stands for the Visual Basic Integrated Design Environment. Graphical Designers
Code Designers
IntelliSense
The Toolbox
Solution Explorer
Properties Window
Server Explorer

**7. Mention any 4 types of graphical designers.**

Windows form designers
Web form designers
Component designers
XML designers

**8. Write the purpose of Properties Window and Form Designer.**

You set properties of various objects in Visual Basic to customize them; for example, we've set the **Text** property of a button in the WinHello project to "Click Me" to make that text appear in the button. To set an object's properties when you're designing your program in Visual Basic-called *design time* -you select that object, and then set the new property values you want in the Properties window.

The Properties window is divided into two columns of text, with the properties on the left, and their settings on the right.

When you're working on a project that has user interface elements-such as forms, VB .NET can display what those elements will look like at run time, and, of course, that's what makes Visual Basic *visual*. For example, when you're looking at a Windows form, you're actually looking at a *Windows form designer*, and you can manipulate the form, as well as add controls to it and so on.

**9. What do you mean by Intellisence?**

IntelliSense is what's responsible for those boxes that open as you write your code, listing all the possible options and even completing your typing for you. IntelliSense is one of the first things you encounter when you use VB .NET.

**10. List any four intellisence features.**

IntelliSense is made up of a number of options, including:

- *List Members*-Lists the members of an object.
- *Parameter Info*-Lists the arguments of procedure calls.

- *Quick Info*-Displays information in tool tips as the mouse rests on elements in your code.
- *Complete Word*-Completes typed words.
- *Automatic Brace Matching*-Adds parentheses or braces as needed.

**11. Name any four tools available in the VB.NET ToolBox.**

Button
TextBox
Label
RadioButton
COmboBox

**12. What is Server Explorer?**

You use the Server Explorer, to explore what's going on in a server. You can do more than just look using the Server Explorer too-you can drag and drop whole items onto Windows forms or Web forms from the Server Explorer.

**13. What is the use of Option Explicit?**

**Option Explicit—** Set to **On** or **Off**. **On** is the default. Requires declaration of all variables before they are used (this is the default).

**14. Write any FOUR unique properties of form.**

- **ControlBox (True or False) :**Forms usually come with minimizing and maximizing buttons, as well as a close box at upper right. To remove these buttons, you can set the form's **ControlBox** property to **False**.
- You can also remove the **minimizing and maximizing** buttons independently, with the **MaximizeBox** and **MinimizeBox** properties.
- The main form, **Form1**, will be our MDI container or parent, containing MDI children, so set its **IsMdiContainer** property to **True**.

**15. Differentiate visible and enable properties.**

**Visible** means that the **property** can be used, but is not **visible** in the graphical user interface. **Enabled/Disable** means that for instance if the object is Disabled, it's still **visible** but the user cannot interact with it.

**16. List any 4 data conversion functions.**

**CBool—** Convert to **Bool** data type.
**CByte—** Convert to **Byte** data type.
**CChar—** Convert to **Char** data type.
**CDate—** Convert to **Date** data type.
**CDbl—** Convert to **Double** data type.
**CDec—** Convert to **Decimal** data type.

**CInt—** Convert to **Int** data type.
**CLng—** Convert to **Long** data type.
**CObj—** Convert to **Object** type.
**CShort—** Convert to **Short** data type.
**CSng—** Convert to **Single** data type.
**CStr—** Convert to **String** type.

**17. Write the syntax of declaring variables and give one example for declaring variable.**

```
Syntax : [ <attrlist> ] [{ Public | Protected | Friend | Protected Friend |
Private | Static }] [ Shared ] [ Shadows ] [ ReadOnly ] Dim [ WithEvents ]
name[ (boundlist) ] [ As [ New ] type ] [ = initexpr ]
Example :
```

Dim EmployeeID As Integer = 1
Dim EmployeeName As String = "Bob Owens"
Dim EmployeeAddress As String

**18. Write the purpose of the following: (1 Mark each)**
**a. IsArray( )**
Returns **True** if passed an array
**b. IsDate( )**
Returns **True** if passed a date
**c. IsDBNull( )**
Returns **True** if passed a database **NULL** value; that is, a **System.DBNull** value
d. IsError( )
Returns **True** if passed an error value
**e. IsNumeric()**
Returns **True** if passed an numeric value
**f. IsReference( )**
Returns **True** if passed an **Object** variable that has no object assigned to it; otherwise, returns **False**

**19. How do you declare arrays in VB.NET? Give example.**
You usually use the **Dim** statement to declare a standard array; here are a few examples of standard array declarations:

```
Dim Data(30)
Dim Strings(10) As String
Dim TwoDArray(20, 40) As Integer
Dim Bounds(10, 100)
```

**20. What is the need of ReDim and Preserve keywords?**
Dynamic arrays can be dimensioned or redimensioned as you need them with the **ReDim** statement. Here's how you use **ReDim**:

```
ReDim [Preserve] varname(subscripts)
```

You use the **Preserve** keyword to preserve the data in an existing array when you change the size of the last dimension.

**21. What are the two ways of writing comments in VB.NET? Give example.**

Comments in Visual Basic start with an apostrophe (') and make Visual Basic ignore whatever follows the apostrophe on the line.

```
NumberStudents = 3      'Three students
        'Display the average value
```

**22. Write the syntax of MID(). Give example.**

**Mid** function to get a substring from the middle of another string if I pass it that string, the location to start extracting the substring from (starting a position 1), and the length of the substring.

Syntax : Mid(string, starting_pos, length)

Example :

```
Dim strText1 As String = "welcome to visual basic"
strText2 = Mid(strText1, 6, 4)
```

**23. Differentiate OrElse and AndAlso logical operators.**

**OrElse - Operator**

This is used to connect two (or more) logical conditions checking like if (condition1 OrElse condition2).

If the first condition is true, It won't perform the second condition. So our execution time is saving in a logical operation in which more conditions are combined using "OrElse" operator.

**AndAlso - Operator**

This is used to connect two (or more) logical conditions checking like if (condition1 AndAlso condition2).

If the first condition is false, It won't perform the second condition . So our execution time is saving in a logical operation in which more conditions are combined using "AndAlso" operator.

**24. Differentaite string operators '&' and '+'**

The + Operator has the primary purpose of adding two numbers. However, it can also concatenate numeric operands with string operands. The + operator has a complex set of rules that determine whether to add, concatenate, signal a compiler error, or throw a run-time InvalidCastException exception.

The & Operator is defined only for String operands, and it always widens its operands to String, regardless of the setting of Option Strict. The & operator is recommended for string concatenation because it is defined exclusively for strings and reduces your chances of generating an unintended conversion.

**25. Differentiate arithmetic operators \ and / with example.**

| | | |
|---|---|---|
| / | It is a division Operator used to divide one operand by another operand and returns a floating-point result. | X / Y |
| \ | It is an integer division Operator, which is similar to division Operator, except that it returns an integer result while dividing one operand to another operand. | X \ Y |

**26. What is the use of With statement. Give an example.**
use the **With** statement to execute statements using a particular object. Here's the syntax:

```
With object
    [statements]
End With
```

Here, I'm use a text box, **Text1**, in a Windows form program, and setting its **Height**, **Width**, and **Text** properties in the **With** statement:

```
With TextBox1
    .Height = 1000
    .Width = 3000
    .Text = "Welcome to Visual Basic"
End With
```

27. Write any two methods of System.Math namespace.
**System.Math.Abs**
**System.Math.Round**
**System.Math.Sqrt**
**System.Math.Exp**

**29. Write the purpose of the following math methods.**
**a. Abs**
Yields the absolute value of a given number.
**b. Atn**
Yields a Double value containing the angle whose tangent is the given number.
**c. Cos**
Yields a Double value containing the cosine of the given angle.
**d. Log**
Yields a Double value containing the logarithm of a given number.
**e. Round**
Yields a Double value containing the number nearest the given value.
**f. Sin**
Yields a Double value specifying the sine of an angle.
**g. Sqr**
Yields a Double value specifying the square root of a number.

**30. Mention the use of any 2 date and time properties.**

| | |
|---|---|
| Get the current date or time | **Today, Now, TimeofDay, DateString, TimeString** |
| Set the date or time | **Today**, **TimeofDay** |
| Return a time | **TimeSerial, TimeValue** |

# UNIT II

**31. Differentiate Sub procedures and functions.**

Sub procedures do not return a value, while functions return a value
the keyword **Function** instead of **Sub,** and specify the return type used in functions.

**32. What is block scope?**

a block is a series of statements terminated by an **End**, **Else**, **Loop**, or **Next** statement, and an element declared within a block can be used only within that block.

- *Block scope*—available only within the code block in which it is declared

**33. What are exceptions?**

Exceptions are just runtime errors; Exceptions occur when a program is running. You can trap such exceptions and recover from them, rather than letting them bring your program to an inglorious end.

**34. What is exception handling in VB? List two types of exception handling.**

Exceptions are just runtime errors; in Visual Basic (unlike some other languages), the terms *exception handling* and *error handling* have become inter-changeable. Exceptions occur when a program is running.

There are two ways of handling errors that occur at run time in VB .NET—with structured and unstructured exception handling.

**35. Write the purpose of Resume Next statement**

**Resume Next** to resume execution with the statement after the one that caused the exception, and **Resume *line***, where *line* is a line number or label that specifies where to resume execution.

```
Sub Main()
        Dim int1 = 0, int2 = 1, int3 As Integer
        On Error Goto Handler
        int3 = int2 / int1
        System.Console.WriteLine("Program completed...")
        Exit Sub
Handler:
        If (TypeOf Err.GetException() Is OverflowException) Then
            System.Console.WriteLine("Overflow error!")
            Resume Next
        End If
    End Sub
```

**36. Give an example for Finally keyword.**

The code in the **Finally** block, if there is one, is always executed in a **Try…Catch…Finally** statement, even if there was no exception, and even if you execute an **Exit Try** statement. This allows you to deallocate resources and so on;

```
Sub Main()
        Dim int1 = 0, int2 = 1, int3 As Integer
        Try
            int3 = int2 / int1
            System.Console.WriteLine("The answer is {0}", int3)
        Catch e As System.OverflowException
            System.Console.WriteLine("Exception: Arithmetic overflow!")
        Catch e As System.ArgumentException
            System.Console.WriteLine("Exception: Invalid argument value!")
        Catch e As System.ArgumentOutOfRangeException
            System.Console.WriteLine("Exception: Argument out of range!")
        Finally
            System.Console.WriteLine("Execution of sensitive code " & _
                "is complete")
        End Try
    End Sub
End Module
```

**37. How do you determine the information on exceptions using Err object?**

The **Err** object also has a new **GetException** method that returns an exception object.

**38. Differentiate IsMdiChild and IsMdiContainer properties of Forms.**

The main form, **Form1**, will be our MDI container or parent, containing MDI children, so set its **IsMdiContainer** property to **True**. This alters its appearance from a white client area to a gray one;

**39. What do you mean by events?**

An **event** is a signal that informs an application that something important has occurred. For example, when a user clicks a control on a form, the form can raise a Click **event** and call a procedure that handles the **event**. **Events** also allow separate tasks to communicate.

**40. What do you mean by methods?**

A **method** is an action that an object **can perform**. For example, Add is a **method** of the ComboBox object, because it adds a new entry to a combo box. The following procedure uses the Add **method** to add a new item to a ComboBox.

**41. Write the purpose of Show and Hide methods of VB.NET Forms.**

use the **Show** and **Hide** methods of controls and forms to show and hide them.

For example, when the user clicks **Button1**, we can hide **Form2** and show **Form3** this way:

```
Private Sub Button1_Click(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles Button1.Click
    Form2.Hide()
    Form3.Show()
End Sub
```

**42. Specify any four values of FormBorder style property.**
set a form's border style with its **FormBorderStyle** property; here are the possible values for that property:

- **Fixed3D—** A fixed, three-dimensional border.
- **FixedDialog—** A thick, fixed dialog-style border.
- **FixedSingle—** A fixed, single-line border.
- **FixedToolWindow—** A tool window border that is not resizable.
- **None—** No border.
- **Sizable—** A resizable border.
- **SizableToolWindow—** A resizable tool window border.

**43. What is the use of Windowstate property? Specify the values .**
sets a value that indicates whether form is minimized, maximized, or normal.
A FormWindowState that represents whether form is minimized, maximized, or normal. The default is FormWindowState.Normal.

**44. What is MDI ?**
You use Multiple Document Interface (MDI) forms. MDI frame windows can display multiple child windows inside them;
The main form, **Form1**, will be our MDI container or parent, containing MDI children, so set its **IsMdiContainer** property to **True**.

**45. Write the MDILayout enumeration values for arranging MDI child forms.**
**MdiLayout** enumeration values that defines the layout of MDI child forms:

- **ArrangeIcons—** All MDI child icons (which are displayed when you minimize an MDI child window) are arranged.
- **Cascade—** All MDI child windows are cascaded.
- **TileHorizontal—** All MDI child windows are tiled horizontally.
- **TileVertical—** All MDI child windows are tiled vertically.

**46. Differentiate GotFocus and LostFocus events.**

Use the **GotFocus** event to specify actions to occur when an object receives the focus. For example, by attaching a **GotFocus** event to each control on a Form, you can guide a user by displaying brief instructions or status bar messages. You can also provide visual cues by enabling, disabling, or showing other controls that depend on the control that has the focus.

The **LostFocus** event occurs after the Exit event. If you move the focus to a control on a form, and that control doesn't have the focus on that form, the Exit and **LostFocus** events for the control that does have the focus on the form occur before the Enter and GotFocus events for the control you moved to.

# UNIT III

**47. How do you make a textbox non editable during run time?**

Use the **ReadOnly** property to make a text box read-only. Setting this property to **True** means that the user cannot enter text into the text box.

**48. Differentiate textbox and rich textbox.**

- Windows forms text boxes are used to get input from the user or to display text. The **TextBox** control is generally used for editable text.
- Use a rich text box. Not only can you enter formatted text (selecting fonts, italics, bolding, and more) in a rich text box, you also can save that text in rich text format (RTF) files, and read RTF files in it.

**49. Differentiate textbox and labels?**

Labels display read-only text, and they give the appearance of text directly on the form; this can look much better than a text box on occasion.

TextBox is box-like controls in which you can enter text. Text boxes can be multiline, have scroll bars, be read-only, and have many other attributes.

**50. Write the code to create a TextBox.**

```
Private Sub Button1_Click(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles Button1.Click
    Dim TextBox1 As New TextBox()
    TextBox1.Size = New Size(150, 20)
    TextBox1.Location = New Point(80, 20)
    TextBox1.Text = "Hello from Visual Basic"
    Me.Controls.Add(TextBox1)
End Sub
```

**51. What is the use of Autosize property of label?**

When you add labels to a form, you can make the label match the text's size by setting the **AutoSize** property to **True**.

**52. How can you link the Web using Link Labels?**

You can recover the text "www.coriolis.com" from the clicked hyperlink with the **e.Link.LinkData.ToString** method, so you open the user's default browser and navigate to that URL this way.

```
System.Diagnostics.Process.Start(e.Link.LinkData.ToString())
```

**53. Differentiate Checkboxes and Radio Buttons.**

- In a ***CheckBox*** group, a user can select more than one option. Each ***CheckBox*** operates individually, so a user can toggle each response "on" and "off".
- ***RadioButton***, however, operates as a group and provide mutually exclusive selection values. A user can select only one option in a radio button group. In other words, clicking

a non-selected radio button will deselect whatever another button was previously selected in the ***RadioButton*** group.

- If you want to enable your user to select more than one option in a group of choices, use C*heckBox'es*. If you want to restrict your user to one option, use R*adioButton's*.

**54. Differentiate Panels and Group boxes.**

The primary **difference** between these two controls is that **GroupBoxes** can display a caption (i.e., text) and do not include scrollbars, whereas **Panels** can include scrollbars and do not include a caption.

**55. Differentiate Visible and Enabled properties.**

Visible means that the property can be used, but is not visible in the graphical user interface. But, if the user were to click a control that edit the objects properties, the object would change even if it isn't visible.

Enabled/Disable means that for instance if the object is Disabled, it's still visible but the user cannot interact with it. This means, that if code is executed for the control that is disabled, nothing will change to that control.

**56. Differentiate Readonly and Enabled properties of textbox.**

**Enabled** specifies whether user interaction is allowed. If a control is disabled then it will not generate any UI events. **ReadOnly** determines whether the user can edit the contents of the control. For example, a **ReadOnly TextBox** cannot be edited, but you can still click on it, select the text contained within it, etc.

**57. What is the use of HideSelection property of the text box.**

While on the topic of text selection, we might note the **HideSelection** property, which, when **True**, turns off text selection highlighting when your program loses the focus.

**58. Write the code for underlining and setting text color of a selected text in a rich textbox.**

**Underlining**

```
RichTextBox1.SelectionStart = RichTextBox1.Find("underlined")
    Dim UnderlineFont As New Font(RichTextBox1.Font, FontStyle.Underline)
    RichTextBox1.SelectionFont = UnderlineFont
```

**Setting text color**

```
RichTextBox3.SelectionStart = RichTextBox3.Find("red")
    RichTextBox3.SelectionColor = Color.Red
```

**59. Specify the various ways of aligning the text in labels.**

- **BottomCenter—** Vertically aligned at the bottom, and horizontally aligned at the center.
- **BottomLeft—** Vertically aligned at the bottom, and horizontally aligned on the left.
- **BottomRight—** Vertically aligned at the bottom, and horizontally aligned on the right.

- **MiddleCenter—** Vertically aligned in the middle, and horizontally aligned at the center.
- **MiddleLeft—** Vertically aligned in the middle, and horizontally aligned on the left.
- **MiddleRight—** Vertically aligned in the middle, and horizontally aligned on the right.
- **TopCenter—** Vertically aligned at the top, and horizontally aligned at the center.
- **TopLeft—** Vertically aligned at the top, and horizontally aligned on the left.
- **TopRight—** Vertically aligned at the top, and horizontally aligned on the right.

**60. How do you create Access Character for a button?**

Just place an **ampersand (&)** in front of the character in the button's caption. For example, placing & in front of the word Me in Click Me (i.e., Click &Me), makes M the access character for that button.

**61. How to access and set the state of a checkbox/radio button? Give example.**

## Getting a Checkbox's State

```
If CheckBox1.Checked Then
        Button1.Text = "The check mark is checked"
    End If
```

## Setting a Checkbox's State

You can set a checkbox's state by setting its **Checked** property to **True** or **False**, as in this code:

```
CheckBox1.Checked = True
```

**62. Mention the use of FlatStyle and Image properties of Button control**

You can add images to buttons at design time and at run time. At design time, you only need to set the **Image** property in the Properties window to an image file. At run time, you can do the same thing if you use the **Image** class's **FromFile** class method and assign the resulting **Image** object to the button's **Image** property.

Setting the button's **FlatStyle** property to **FlatStyle.Flat** to make it appear flat

**63. How do you add picture to a button.**

You can add images to buttons at design time and at run time. At design time, you only need to set the **Image** property in the Properties window to an image file. At run time, you can do the same thing if you use the **Image** class's **FromFile** class method and assign the resulting **Image** object to the button's **Image** property.

```
Button1.Image = Image.FromFile("C:\vbnet\ch06\imagebuttons\clicked.jpg")
```

**64. Describe the items and multicoloumn properties of a ListBox.**

At design time, you can use the **Items** property, which is a very handy array of the items in the list box, and at run time, you can use both the **Items** property and the **Add** (formerly **AddItem**) method.

If you set the **MultiColumn** property of a list box to **True**, giving it multiple columns.

**65. How do you determine the selected item in a list/combo box?**

If a list box supports only single selections, the **SelectedItem** property to get the selected item itself.

```
TextBox1.Text = ListBox1.SelectedIndex
```

**66. Describe the sorted, and Text properties of a listbox.**

Sorting a List Box

You can alphabetize the items in a list box by setting its **Sorted** property to **True** (it's **False** by default) at design time or run time. That's all it takes, just set its **Sorted** property to **True**.

| | |
|---|---|
| **Text** | Selects the text of the selected item in the list box. |

```
TextBox1.Text = ListBox1.Text
```

**67. Write the purpose of CheckedIndeces and GetItemCheckState properties of Checkedlist box.**

a **CheckedIndices** property, which returns a collection holding the indices of the checked items in the checked list box.

use **GetItemCheckState** to determine the check *state* of an item.

**68. How to**

**a. Clear a a combo box**

You can clear a whole combo box at once with the **Items.Clear** method.

```
ComboBox1.Items.Clear()
```

**b. Get the number of items in a combo box.**

You can get the number of items in a combo box with the **Items.Count** property like this:

```
MsgBox("The combo box contains " & ComboBox1.Items.Count & " items.")
```

**69. Differentiate Listbox and combobox.**

**List Box :**

1. Occupies more space but shows more than one value.
2. We can select multiple items.
3. we can use checkboxes with in the list box.

**Combo Box:**

1. Occupies less space but shows only one value for visibility
2. Multiple select is not possible
3. can't use checkboxes within combo boxes

**70. How to create multiselect list boxes? Give example.**

multiselect list box at run time. To make the list box a multiselect list box, I'll set its **SelectionMode** property to **MultiExtended**;

```
ListBox1.SelectionMode = SelectionMode.MultiExtended
  ListBox1.SetSelected(1, True)
    ListBox1.SetSelected(3, True)
    ListBox1.SetSelected(5, True)
```

**71. Mention the purpose of Progressbar and Trackbar.**

**Progress bars** are those simple controls that show the progress of some operation by displaying rectangles in a horizontal bar.

**Track bars** are very much like scroll bars but differ in appearance. Track bars look more like controls you might see on stereos.

**72. How to add collection of objects in a listbox at once?**

You also can use the **AddRange** method to add a collection of objects to a list box all at once;

```
Dim DataArray(4) As String
    Dim intLoopIndex As Integer

    For intLoopIndex = 0 To 4
        DataArray(intLoopIndex) = New String("Item " & intLoopIndex)
    Next intLoopIndex

    ListBox1.Items.AddRange(DataArray)
```

**73. Write the purpose of Date, Day, DayOfWeek, Ticks properties of DateTime Picker.**

**74. What is the purpose of ToolTip? How to set it?**

They're those small windows that appear with explanatory text when you let the mouse rest on a control or window.

You can associate a tool tip with any other control. To connect a tool tip with a control, you use its **SetToolTip** method. For example, to connect the tool tip you see in Figure 8.6 to **Button1**, you can use this code:

```
ToolTip1.SetToolTip(Button1, "This is a button")
```

**75. Differentiate DateTimePicker and MonthCalender Controls.**

**76. Name FOUR built in Dialog Boxes.**

- Open File dialogs
- Save File dialogs
- Font dialogs
- Color dialogs
- Print Preview dialogs
- Page Setup dialogs
- Print dialogs

**77. Explain Menu Access Keys and Creating Menu Shortcuts.**

Access keys make it possible to select menu items from the keyboard using the Alt key.
To give an item an access key, you precede the access key in its caption with an ampersand (&).

You can create shortcuts for menu items, which are key combinations that, when pressed, will select that item, making its **Click** event happen. You set a shortcut with the **Shortcut** property. To display the shortcut next to the menu item's caption at run time, you set the **ShowShortcut** property to **True** (it's **True** by default).

```
menuItem1.Shortcut = Shortcut.CtrlX
```

**78. Mention Four modes of List Views.**

You can use a **list view** to create a user interface like **the** right pane of **Windows** Explorer. **The** control has **four view modes**: LargeIcon, SmallIcon, **List**, and Details.

## UNIT IV

**79. How to add checkmark to menu items using Menustrip control?**

To add a checkmark to a menu item at design time, just click to the left of its caption.
You can use the **Checked** property of a **MenuItem** object to toggle the checkmark; **True** means the checkmark is displayed, **False** means it's hidden.

```
MenuItem7.Checked = Not MenuItem7.Checked
```

**80. How to add access key to menu items in Menustrip control?**

Access keys make it possible to select menu items from the keyboard using the Alt key.
To give an item an access key, you precede the access key in its caption with an ampersand (&). In this example, that means using the captions "&File" and "E&xit". Access keys are underlined in menu captions.

**81. How to create shortcuts to access menu items in Menu control?**

You can create shortcuts for menu items, which are key combinations that, when pressed, will select that item, making its **Click** event happen. You set a shortcut with the **Shortcut** property.

To assign a shortcut key combination to a menu item at design time, just select a shortcut key combination from the list that appears when you select the **Shortcut** property of any menu item in the properties window. At run time, you can use members of the **Shortcut** enumeration to do the same thing, as in this code:

```
menuItem1.Shortcut = Shortcut.CtrlX
```

**82. What are the steps involved to merge menu items?**

To merge these menus,

I set the **MergeType** property of the MDI child's Edit menu to **MergeItems**,
and set the **MergeType** property of the Copy item in that menu to **Add**.
Then I set the **MergeType** of the Edit menu in the MDI parent to **MergeItems** and the **MergeType** property of the Cut item in that menu to **Add**.

**83. What is context menu? How to use it?**
You use **ContextMenu** controls to give users access to frequently used menu commands, and bring them up by right-clicking another control.

You associate context menus with other controls by setting the control's **ContextMenu** property to the **ContextMenu** control. The central property of the **ContextMenu** control is the **MenuItems** property; you can add menu items to a context menu at design time or in code by creating **MenuItem** objects and adding them to the **MenuItems** collection of the context menu.

**85. List any 4 properties of ColorDialog control class.**
**AllowFullOpen**
**SolidColorOnly**
**CustomColors**
**SolidColorOnly**

**86. List any 4 properties of SaveFileDialog control class.**
**AddExtension**
**InitialDirectory**
**Filter**
**DefaultExt**

**87. What is the purpose of ADO.NET?**
ADO.NET is the main data access system and protocol that Visual Basic .NET uses. As we've already mentioned, it uses a disconnected data architecture, which means that the data you work with is just a copy of the data in the data in the actual database.

**88. Difference ExecuteReader() and ExecuteScalar() methods.**
**ExecuteReader—** Executes SQL commands that return rows. Note that this method does *not* return a dataset, it creates a data reader, which is much more simplistic.

**ExecuteScalar—** Calculates and returns a single value, such as a sum, from a database.

**89. What is data adapter? List any two data adapters in VB.NET.**

*Data adapters*—Data adapters are a very important part of ADO.NET. You use them to communicate between a data source and a dataset. You typically configure a data adapter with SQL to execute against the data source. The two types of data adapters are **OleDbDataAdapter** and **SqlDataAdapter** objects.

**90. What are the Steps involved while accessing data in dataset.**

1. Create a data connection or use an existing data connection.
2. Drag an **OleDBAdaptor** or **SQLAdaptor** object onto a form; creates connection and adaptor objects.
3. Use the Data Adapter Configuration Wizard to configure the data adapter and create the SQL you want.
4. Generate a dataset.
5. Bind the dataset to controls.
6. Fill the dataset in code.

**91. Write a note on Command object in ADO.**

*Command objects*—Data adapters can read, add, update, and delete records in a data source. To allow you to specify how each of these operations work, a data adapter contains command objects for each of them. Data adapters support four properties that give you access to these command objects: **SelectCommand**, **InsertCommand**, **UpdateCommand**, and **DeleteCommand**.

**92. What is data binding? List different types of data binding available in VB.NET**

This usually means binding controls to data from databases; for example, you might bind a text box to the last names of authors (that is, the au_lname field) from the **authors** table of the pubs database, which would make the text box display those names automatically as you moved through the database.

Simple Binding

Complex Binding

**93. Write the difference of Simple Data Binding and Complex Data Binding.**

Simple data binding binds to one data item at a time, such as a name displayed in a text box, but complex data binding allows a control to bind to more than one data element, such as more than one record in a database, at the same time.

94. What are the advantages of data binding?

Some of the advantages of data binding are listed below:

1. Data based application can be created quickly and efficiently.
2. Coding size decreases incredibly still you get the desired result.
3. Execution time increases and hence it increases the quality of the application.
4. By using events you can gain control over data binding process.
5. Even if your data is bounded you can modify the coding part easily.
6. Programmers get the benefit of both unbound and bound approach while creating any application.

95. What are the disadvantages of data binding.

Some of the disadvantages of data binding are listed below:

1. Optimization of code is only achieved by using traditional or unbound approach.

2. Data binding is not completely flexible and for achieving more flexibility traditional methods are used by many programmers.

96. How to list all records from a table using SQL statement?

- Select * from table_name