'1. find area and circumference of circle (Using Variable, constant, and arithmetic operator)

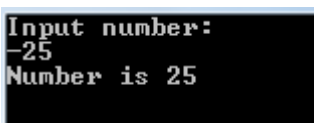
```
Option Strict On ' enable type checking
Option Explicit On ' enable variable declaration
Module Module1
   Sub main()
      Dim r As Single
      Dim a, c As Double
      Const pi As Single = 22 / 7
      Console.WriteLine("Input radius of circle:")
      r = CSng(Console.ReadLine())
      a = pi * r * r
      c = 2 * pi * r
      Console.WriteLine("Area={0},Circumference={1}", a, c)

      Console.Read()
   End Sub
End Module
```

Output:

'2. Convert input number in positive (Using If)

```
Module Module2
   Sub main()
      Dim n As Double
      Console.WriteLine("Input number:")
      n = CDbl(Console.ReadLine())
      If n < 0 Then
         n = -n
      End If
      Console.WriteLine("Number is {0}", n)
      Console.Read()
   End Sub
End Module
```

Output:

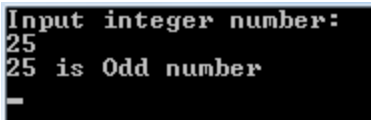


'3. Find input number is even or odd (Using If-Else)

```
Module Module3
   Sub main()
      Dim n As Integer
      Console.WriteLine("Input integer number:")
      n = CInt(Console.ReadLine())
```

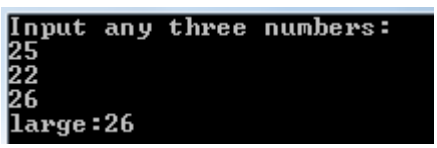
```
        If n Mod 2 = 0 Then
            Console.WriteLine("{0} is Even number", n)
        Else
            Console.WriteLine("{0} is Odd number", n)
        End If
        Console.Read()
    End Sub
End Module
```

Output:




```
'4. find larger number among three (Using nested If-Else)
Module Module4
    Sub main()
        Dim a, b, c As Single
        Console.WriteLine("Input any three numbers:")
        a = CSng(Console.ReadLine())
        b = CSng(Console.ReadLine())
        c = CSng(Console.ReadLine())

        If a > b Then
            If a > c Then
                Console.WriteLine("large:"&a)
            Else
                Console.WriteLine(("large:"&c)
            End If
        Else
            If b > c Then
                Console.WriteLine(("large:"&b)
            Else
                Console.WriteLine(("large:"&c)
            End If
        End If

        Console.Read()
    End Sub
End Module
```

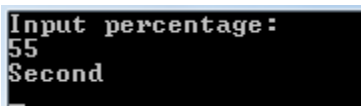
Output:



'5. Show division of input percentage (Using Elseif ladder)

```
Module Module5
   Sub main()
      Dim p As Single
      Console.WriteLine("Input percentage:")
      p = CSng(Console.ReadLine())
      Dim div As String
      If p >= 60 Then
         div = "First"
      ElseIf p >= 45 Then
         div = "Second"
      ElseIf p >= 33 Then
         div = "Third"
      Else
         div = "Fail"
      End If
      Console.WriteLine(div)
      Console.Read()
   End Sub
End Module
```

Output:



' 6. Arithmetic Operation on selection base(Using Select Case)

```
Module Module6
   Sub main()
      Dim a, b As Integer
      a = 10
      b = 3
      Dim n As Integer
      n = 6
      Select Case n
         Case 1
            Console.WriteLine(a + b)
         Case 2
            Console.WriteLine(a - b)
         Case 3
            Console.WriteLine(a * b)
         Case 4
            Console.WriteLine(a / b)
         Case 5
            Console.WriteLine(a \ b)
         Case 6
            Console.WriteLine(a Mod b)
         Case Else
            Console.WriteLine("Invalid selection")
      End Select
      Console.Read()
```

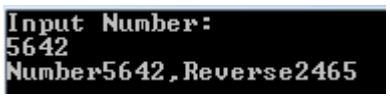
```
    End Sub
End Module
```

Output:

'7. Reverse Input number (Using while loop)

```
Module Module7
    Sub main()
        Dim m, n, r As Integer
        Console.WriteLine("Input Number:")
        m = CInt(Console.ReadLine())
        n = m
        r = 0
        While n > 0
            r = r * 10 + n Mod 10
            n \= 10
        End While
        Console.WriteLine("Number{0},Reverse{1}", m, r)

        Console.Read()
    End Sub
End Module
```

Output:



'8. Input number is prime or not (Using For loop)

```
Module Module8
    Sub main()
        Dim n, i As Integer
        Console.WriteLine("Input Number:")
        n = CInt(Console.ReadLine())
        For i = 2 To n - 1
            If n Mod i = 0 Then Exit For
        Next
        If i = n Then
            Console.WriteLine("{0} is Prime", n)
        Else
            Console.WriteLine("{0} is not Prime", n)
        End If

        Console.Read()
    End Sub
End Module
```

Output:

```
Input Number:
17
17 is Prime
```

'9. Read 5 numbers and print all with sum. (Using one dimensional array)

```
Module Module9
   Sub main()
      Dim arr(5) As Integer
      Dim i As Integer
      Console.WriteLine("Input 5 Numbers:")
      arr(0) = 0  ' for sum of numbers
      For i = 1 To 5
         arr(i) = CInt(Console.ReadLine())
         arr(0) += arr(i)
      Next

      Console.WriteLine("First is sum of all")
      For i = 0 To 5
         Console.WriteLine(arr(i))
      Next

      Console.Read()

   End Sub
End Module
```

Output:

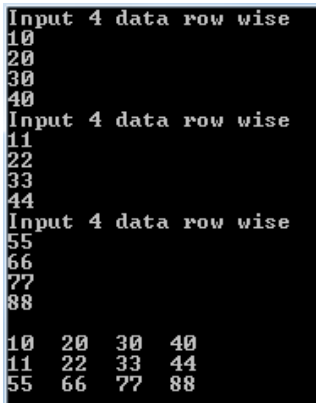
```
Input 5 Numbers:
10
20
30
40
50
First is sum of all
150
10
20
30
40
50
```

'10. Print 3x4 matrix (Using two dimessional array)

```
Module10
   Sub Main()
      Dim m(2, 3) As Integer                    '3x4
      Dim i, j As Integer
      'Read matrix
      For i = 0 To 2
         Console.WriteLine("Input 4 data row wise")
         For j = 0 To 3
            m(i, j) = CInt(Console.ReadLine())
         Next
      Next
```

```
        'print input matrix
        For i = 0 To 2
            Console.WriteLine("")
            For j = 0 To 3
                Console.Write(m(i, j) & "  ")
            Next
        Next

        Console.Read()
    End Sub
End Module
```

Output:

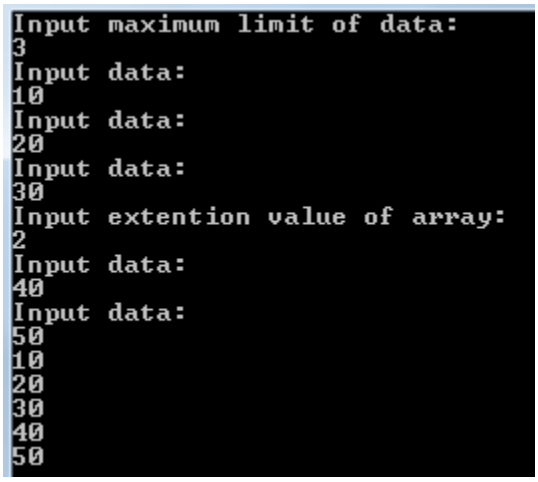


'11. Dynamic array (Using Redim, Preserve keyword)

```
Module11
    Sub main()
        Dim darr() As Integer
        Console.WriteLine("Input maximum limit of data:")
        Dim n As Integer = CInt(Console.ReadLine())
        ReDim darr(n)

        Dim i As Integer
        For i = 1 To n
            Console.WriteLine("Input data:")
            darr(i) = CInt(Console.ReadLine())
        Next

        ' To extend dynamic array with old data
        Console.WriteLine("Input extention value of array:")
        Dim m As Integer = CInt(Console.ReadLine())
        ReDim Preserve darr(n + m)
        For i = n + 1 To n + m
            Console.WriteLine("Input data:")
            darr(i) = CInt(Console.ReadLine())
        Next
        'print all data of dynamic array
        For i = 1 To n + m
            Console.WriteLine(darr(i))
```

```
        Next
        Console.Read()

    End Sub
End Module
```

Output:



'12. Sub routine with call by value and call by reference

```
Module Module12
    Sub LG(ByVal a As Integer, ByRef b As Single)
        a = 10
        b = 22.5
    End Sub

    Sub main()
        Dim x As Integer
        Dim y As Single
        x = 20
        y = 11.5
        Console.WriteLine("Before calling procedure: x=" & x & " y=" & y)
        LG(x, y)

        Console.WriteLine("After calling procedure: x=" & x & " y=" & y)

        Console.Read()
    End Sub
End Module
```

Output:

'13. Function for factorial number using recursion

```
Module Module13
    Function Fact(ByVal n As Integer) As Long
```

```
        Dim f As Long
        If n = 1 Then
            Return 1
        Else
            f = n * Fact(n - 1)       'Recursive process
            Return f
        End If
    End Function

    Sub main()
        Dim ft As Long
        ft = Fact(5)    'calling function
        Console.WriteLine(ft)

        Console.Read()
    End Sub

End Module
```

Output:

'14. Making default argument using Optional keyword

```
Module Module14
    Function UseOpt(ByVal a As Integer, Optional ByVal b As Integer = 0, Optional ByVal c As Integer = 0) As Integer
        Return a + b + c
    End Function
    Sub main()
        Dim r As Integer
        r = UseOpt(10, 20, 30) 'Pass 3 values
        Console.WriteLine(r)

        r = UseOpt(10, 20)   'Pass 2 values
        Console.WriteLine(r)

        r = UseOpt(10)   'Pass 1 values
        Console.WriteLine(r)
        Console.Read()
    End Sub
End Module
```

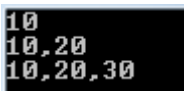
Output:

' 15. Function/method overloading

```
Module Module15
    Sub Show(ByVal a As Integer)
        Console.WriteLine(a)
```

```
    End Sub

    Sub Show(ByVal a As Integer, ByVal b As Integer)
        Console.WriteLine(a & "," & b)
    End Sub

    Sub Show (ByVal a As Integer, ByVal b As Integer, ByVal c As Integer)
        Console.WriteLine(a & "," & b & "," & c)
    End Sub

    Sub main()
        Show(10)  'Pass 1 values
        Show(10, 20) 'Pass 2 values
        Show(10, 20, 30) 'Pass 3 values

        Console.Read()
    End Sub
End Module
```

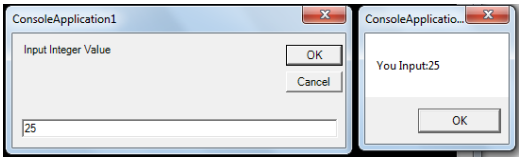
Output:



'16. Using MsgBox and InputBox

```
Module Module16
    Sub main()
        Dim a As Integer
        Dim val As String
        val = InputBox("Input Integer Value")
        a = CInt(val)
        MsgBox("You Input:" & a)
    End Sub
End Module
```

Output:

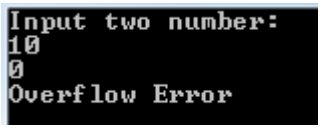


'17. Exception Handling using On Error Handler

```
Module Module17
    Sub main()
        On Error GoTo handler
        Dim a, b, c As Integer
        Console.WriteLine("Input two number:")
        a = CInt(Console.ReadLine())
        b = CInt(Console.ReadLine())
        c = a / b
```

```
        Console.WriteLine(c)
nextline:
        Console.Read()
        Exit Sub
handler:
        Console.WriteLine("Overflow Error")
        Resume nextline
    End Sub
End Module
```

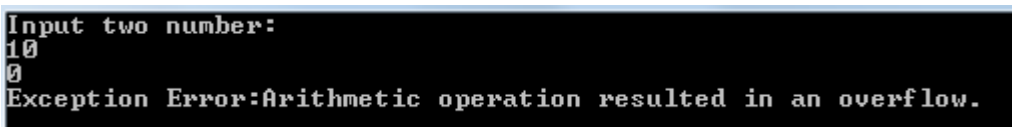
Output:



'18. Exception Handling using Try Catch

```
Module Module18
    Sub main()
        Try
            Dim a, b, c As Integer
            Console.WriteLine("Input two number:")
            a = CInt(Console.ReadLine())
            b = CInt(Console.ReadLine())
            c = a / b
            Console.WriteLine(c)
        Catch ex As Exception
            Console.WriteLine("Exception Error:{0}", ex.Message)
        End Try
        Console.Read()
    End Sub
End Module
```

Output:



'19. Exception Handling used Finally with try and catch

```
Module Module19
    Sub main()
        Try
            Dim a, b, c As Integer
            Console.WriteLine("Input two number:")
            a = CInt(Console.ReadLine())
            b = CInt(Console.ReadLine())
            c = a / b
            Console.WriteLine(c)
        Catch ex As Exception
            Console.WriteLine("{0}:Error", ex.Message)
```

```
        Finally
            Console.WriteLine("Execution code completed")
        End Try
        Console.Read()
    End Sub
End Module
```

Output:

```
Input two number:
10
0
Arithmetic operation resulted in an overflow.:Error
Execution code completed
```

'20. Use Throw to catch even exception made or not

```
Module Module20
    Sub main()
        Try
            Throw Err.GetException
        Catch ex As Exception
            Console.WriteLine("{0}:Error", ex.Message)
        End Try
        Console.Read()
    End Sub
End Module
```

```
Object reference not set to an instance of an object.:Error
```

Output:

'21. Structured exception handling using try catch

```
Module Module21
    Sub main()
        Try
            Dim a, b, c As Integer
            Console.WriteLine("Input two number:")
            a = CInt(Console.ReadLine())
            b = CInt(Console.ReadLine())
            c = a / b
            Console.WriteLine(c)

        Catch ex As Exception
            Console.WriteLine("{0}:Error", ex.Message)
        Finally
            Console.WriteLine("Execution code completed")
        End Try
        Console.Read()
    End Sub
End Module
```

Output:

```
Object reference not set to an instance of an object.:Error
```

'22. Using Class with data, subroutine, function, property as members and creating object and calling method using object

```
Module Module22
  Class A
    Private x As Integer
    Public Sub SR(ByVal a As Integer)
      x = a
    End Sub

    Public Function Fun() As Integer
      Return x
    End Function
    Public Property pr() As Integer
      Get
        Return x
      End Get
      Set(ByVal a As Integer)
        x = a
      End Set
    End Property

  End Class
  Sub main()
    Dim ob As A = New A()
    ob.SR(2)
    Dim var As Integer = ob.Fun()
    Console.WriteLine(var)

    ob.pr = 20
    Console.WriteLine(ob.pr)
    Console.Read()
  End Sub
End Module
```

Output:

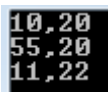
```
2
20
```

'23. Using constructor and destructor

```
Module Module23
  Class A
    Private a As Integer
    Private b As Integer
    Public Sub New()
      a = 10
      b = 20
    End Sub
    Public Sub New(ByVal x As Integer)
```

```
        a = x
        b = 20
    End Sub
    Public Sub New(ByVal x As Integer, ByVal y As Integer)
        a = x
        b = y
    End Sub
    Protected Overrides Sub finalize()
        Console.WriteLine("Object removed")
        Beep()
        Console.Read()
    End Sub

    Public Sub show()
        Console.WriteLine(a & "," & b)
    End Sub
  End Class
  Sub main()
    Dim ob1 As A = New A()
    Dim ob2 As A = New A(55)
    Dim ob3 As A = New A(11, 22)

    ob1.show()
    ob2.show()
    ob3.show()

    Console.Read()
  End Sub
End Module
```

Output:

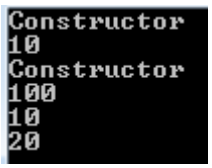


'24. Using constructor and destructor

```
Module Module24
  Public Class ClassX
    Private a As Integer
    Public Property pa() As Integer
      Get
        Return a
      End Get
      Set(ByVal value As Integer)
        a = value
      End Set
    End Property

    Public Sub SR()
      Console.WriteLine(a)
```

```
    End Sub
    Public Function Fun() As Integer
      Return a * 2
    End Function

    Public Sub New()
      Console.WriteLine("Constructor")
    End Sub
    Public Sub New(ByVal x As Integer)
      a = x
      Console.WriteLine(a)
    End Sub
    Protected Overrides Sub finalize()
      Beep()
    End Sub
  End Class

  Sub main()
    Dim ob1 As New ClassX
    Dim ob2 As ClassX = New ClassX(10)
    Dim ob3 As ClassX
    ob3 = New ClassX()
    ob1.pa = 100    'set
    Console.WriteLine(ob1.pa) 'get

    ob2.SR()

    Dim p As Integer = ob2.Fun()
    Console.WriteLine(p)

    Console.Read()
  End Sub
End Module
```

Output:

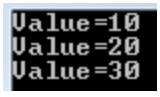


'25. Simple program for inheritance

```
Module Module25
  Class B
    Private a As Integer
    Protected b As Integer
    Public c As Integer
    Public Property pa() As Integer
      Get
        Return a
      End Get
```

```
        Set(ByVal value As Integer)
            a = value
        End Set
    End Property

End Class
Class D
    Inherits B   'Creates Inheritance
    Public Property pb() As Integer   'use protected b of Base
        Get
            Return b
        End Get
        Set(ByVal value As Integer)
            b = value
        End Set
    End Property
End Class
Sub main()
    Dim od As D = New D()
    od.c = 10   'c is public data of Base
    Console.WriteLine("Value={0}", od.c)

    od.pb = 20  'pb is public property of Derived
    Console.WriteLine("Value={0}", od.pb)

    od.pa = 30   'pa is public property of Base
    Console.WriteLine("Value={0}", od.pa)
    Console.Read()
End Sub
End Module
```

Output:

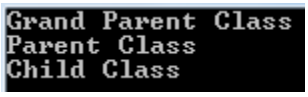


' 26. Constructor in case of inheritance

```
Module Module26
    Class G
        Public Sub New()
            Console.WriteLine("Grand Parent Class")
        End Sub
    End Class
    Class P
        Inherits G
        Public Sub New()
            Console.WriteLine("Parent Class")
        End Sub
    End Class
    Class C
        Inherits P
```

```
        Public Sub New()
            Console.WriteLine("Child Class")
        End Sub
    End Class
    Sub main()
        Dim oc As C = New C()

        Console.Read()
    End Sub
End Module
```

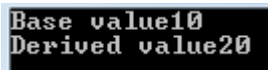
Output:



'27. Passing argument to base class Constructor using derived object

```
Module Module27
    Class Base
        Private a As Integer
        Public Sub New(ByVal x As Integer)
            a = x
            Console.WriteLine("Base value{0}", a)
        End Sub
    End Class
    Class Derived
        Inherits Base
        Private b As Integer
        Public Sub New(ByVal x As Integer, ByVal y As Integer)
            MyBase.New(x)
            b = y
            Console.WriteLine("Derived value{0}", b)
        End Sub
    End Class
    Sub main()
        Dim od As Derived = New Derived(10, 20)

        Console.Read()
    End Sub
End Module
```

Output:



'28. Using Overridable and Overrides

```
Module Module28
    Public Class Base
        Public Overridable Sub show()
            Console.WriteLine("Base show Method")
```

```
    End Sub
  End Class
  Public Class Derived
    Inherits Base
    Public Overrides Sub show()
      Console.WriteLine("Derived show method")
    End Sub
  End Class
  Sub main()
    Dim od As New Derived
    od.show()  'call method of derived

    Console.Read()
  End Sub
End Module
```

Output:


```
'29. calling Overridable method of base using MyClass Keyword
Module Module29
  Public Class Base
    Public Overridable Sub show()
      Console.WriteLine("Base show Method")
    End Sub
    Public Sub callshow()
      show()
    End Sub
    Public Sub callshow1()
      MyClass.show()
    End Sub
  End Class
  Public Class Derived
    Inherits Base
    Public Overrides Sub show()
      Console.WriteLine("Derived show method")
    End Sub

  End Class
  Sub main()
    Dim od As New Derived
    od.callshow()  'derived show
    od.callshow1() 'base show

    Console.Read()
  End Sub
End Module
```

Output:

```
Derived show method
Base show Method
```

'30. Implement concept of polymorphism

```
Module Module30
    Public Class Base
        Public Overridable Sub show()
            Console.WriteLine("Base show Method")
        End Sub
    End Class
    Public Class Derived
        Inherits Base
        Public Overrides Sub show()
            Console.WriteLine("Derived show method")
        End Sub
    End Class
    Sub main()
        Dim ob As New Base
        Dim od As New Derived
        poly(ob)    'pass base object to base variable
        poly(od)    'pass derived object to base variable
        Console.Read()
    End Sub
    Sub poly(ByVal ob As Base) ' gets base or derived object
        ob.show()   'call show method of base or derived
    End Sub
End Module
```

Output:

```
Base show Method
Derived show method
```

'31. using interface

```
Module Module31
    Interface I
        Sub show()
    End Interface

    Class A
        Implements I

        Public Sub show() Implements I.show
            Console.WriteLine("Interface Method defined in class")
        End Sub
    End Class
    Sub main()
        Dim ob As New A
        ob.show()

        Console.Read()
    End Sub
```

```
End Module
```

Output:

```
Interface Method defined in class
```

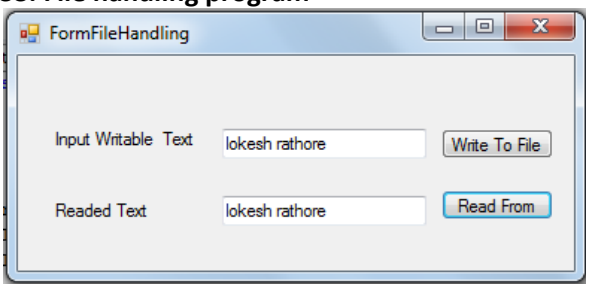
'32. use modifier with inheritence-NotInheritable, MustInherit

```
Module Module32
    'Class B can's be inherits by derived class
    NotInheritable Class A
        Public Sub show()
            Console.WriteLine("I m not inheritable")
        End Sub
    End Class
    'Object of class B cant be created. Only will be inherit
    MustInherit Class B
        Public Sub show()
            Console.WriteLine("I m must inheritable")
        End Sub
    End Class

    Class D
        Inherits B
    End Class
    Sub main()
        Dim oa As New A ' A is NotInheritable
        oa.show() 'call method of A

        ' Dim ob As New B  Here B is MustInherit
        Dim od As New D
        od.show() 'call method of B

        Console.Read()
    End Sub
End Module
```

Output:

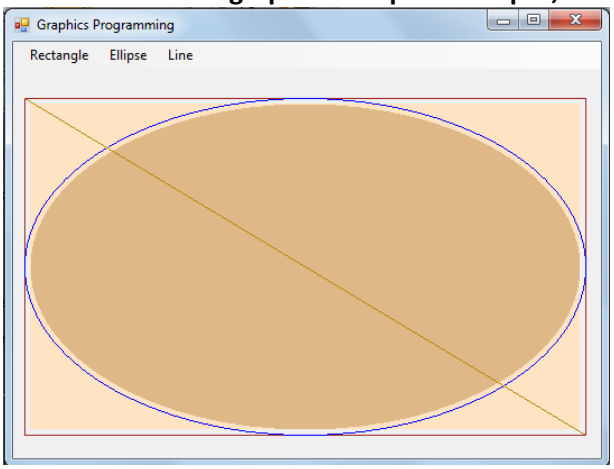
```
I m not inheritable
I m must inheritable
```

33. File handling program



```
Imports System.io
Public Class Form01FileHandling
Private Sub ButtonWrite_Click()
       Dim fw As FileStream = New FileStream("c:\xyz.txt", FileMode.Create,
                                                            FileAccess.Write)
        Dim w As New StreamWriter(fw)
        w.WriteLine(TextBox1.Text)
        w.Flush()
        w.Close()
End Sub
Private Sub ButtonRead_Click()
        Dim fr As FileStream = New FileStream("c:\xyz.txt", FileMode.Open,
                                                                FileAccess.Read)
        Dim r As New StreamReader(fr)
        TextBox2.Text = r.ReadToEnd()
        r.Close()
End Sub
End Class
```

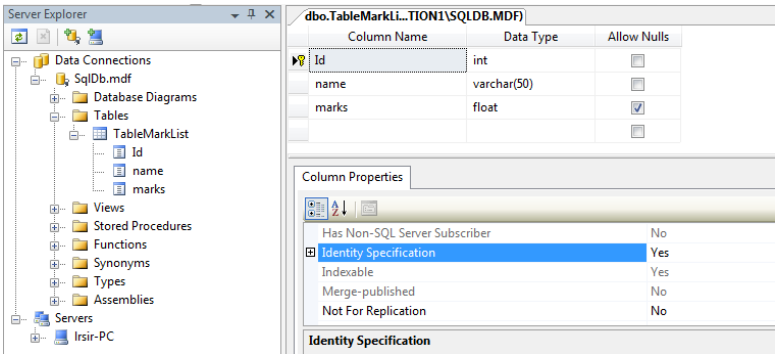
34. Draw different graphical shaps like ellipse, rectangle and line.



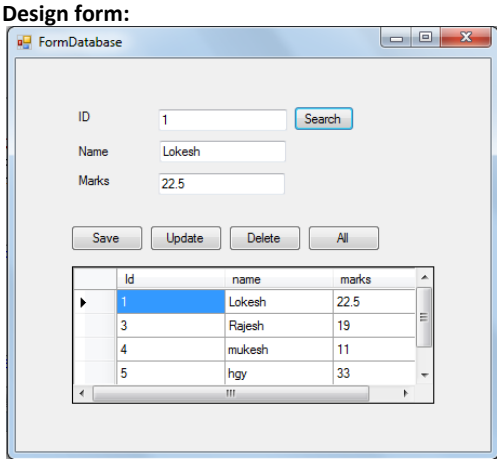

```
Public Class Form02Graphics
    Private Sub EmptyRectangleToolStripMenuItem_Click()
        Dim g As Graphics = Me.CreateGraphics()
        Dim r As New Rectangle(10, 50, 500, 300)
        g.DrawRectangle(Pens.Brown, r)
    End Sub
    Private Sub FilledRectangleToolStripMenuItem_Click()
        Dim g As Graphics = Me.CreateGraphics()
        Dim f As New RectangleF(15, 55, 490, 290)
        g.FillRectangle(Brushes.Bisque, f)
    End Sub
    Private Sub EmptyEllipseToolStripMenuItem_Click()
        Dim g As Graphics = Me.CreateGraphics()
        Dim r As New Rectangle(10, 50, 500, 300)
        g.DrawEllipse(Pens.Blue, r)
    End Sub
    Private Sub FilledEllipseToolStripMenuItem_Click()
        Dim g As Graphics = Me.CreateGraphics()
        Dim f As New RectangleF(15, 55, 490, 290)
        g.FillEllipse(Brushes.BurlyWood, f)
    End Sub
    Private Sub LineToolStripMenuItem_Click()
        Dim g As Graphics = Me.CreateGraphics()
        g.DrawLine(Pens.DarkGoldenrod, 10, 50, 510, 350)
    End Sub

End Class
```

35. Working with database like insert, update, delete and select records using form.

Create database: Using MS SqlServer, Create a database file (SqlDb.mdf) having a table(TableMarkList) with some fileds(Id, name, marks) as below.



Design form:

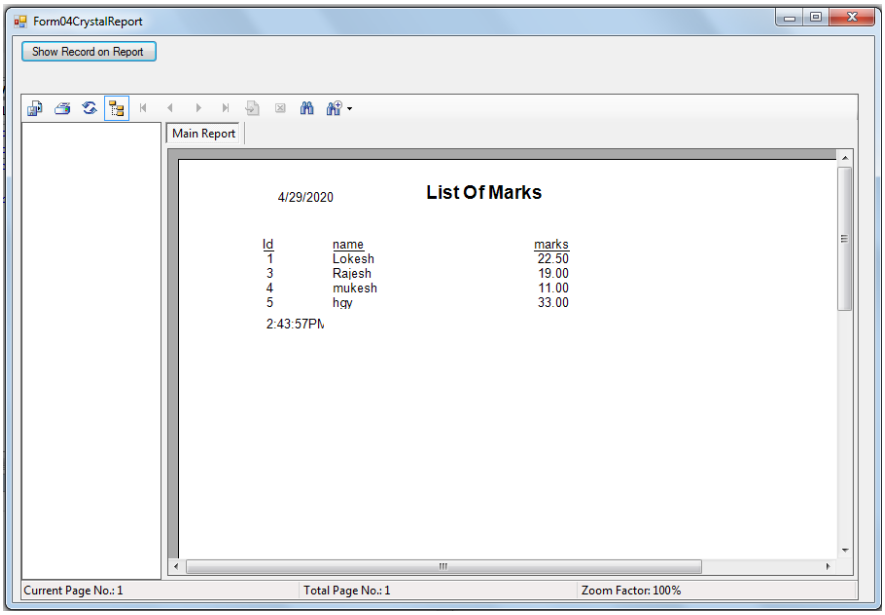


Coding:

```
Imports System.Data.SqlClient
'Database class
Public Class ClassDB
    Public con As SqlConnection
    Public cmd As SqlCommand
    Public dr As SqlDataReader
    Public da As SqlDataAdapter
    Public ds As DataSet

    Public Sub connect()
        Dim constr As String
        constr = "Data Source=.\SQLEXPRESS;AttachDbFilename=|DataDirectory|
                            SqlDb.mdf;Integrated Security=True;User Instance=True"
        con = New SqlConnection(constr)
        If con.State <> ConnectionState.Open Then
            con.Open()
        End If
    End Sub

    Public Sub IUD(ByVal sql As String)
        connect()
        cmd = New SqlCommand(sql, con)
        cmd.ExecuteNonQuery()

    End Sub

    Public Function SearchRecord(ByVal sql As String) As Boolean
        connect()
        cmd = New SqlCommand(sql, con)
```

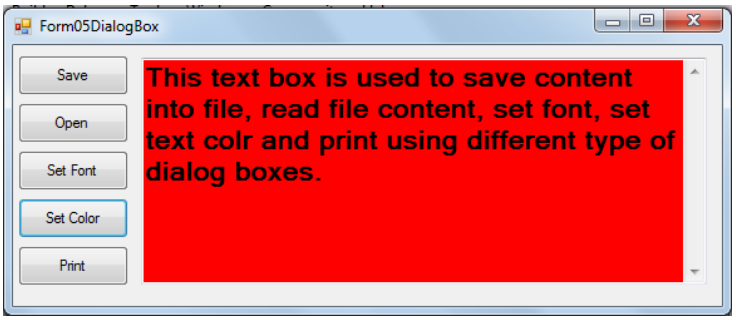
```
        dr = cmd.ExecuteReader()
        If dr.Read Then
            Return True
        Else
            Return False
        End If
    End Function

    Public Sub FillDS(ByVal sql As String)
        connect()
        cmd = New SqlCommand(sql, con)
        da = New SqlDataAdapter(cmd)
        ds = New DataSet()
        da.Fill(ds)
    End Sub
End Class


Public Class Form03Database
    Private Sub ButtonSearch_Click()
        Dim sql As String
        sql = "Select * From TableMarkList Where ID=" & TextBox1.Text
        Dim dbw As New ClassDB()
        Dim found As Boolean
        found = dbw.SearchRecord(sql)
        If found Then
            TextBox2.Text = dbw.dr(1)
            TextBox3.Text = dbw.dr(2)
        Else
            MsgBox("Not Found")
        End If
    End Sub

    Private Sub ButtonInsert_Click()
        Dim sql As String
        sql = "Insert Into TableMarkList(name,marks) Values('" & TextBox2.Text &
                                                        "'," & TextBox3.Text & ")"
        Dim dbw As New ClassDB()
        dbw.IUD(sql)
        MsgBox("Saved")
    End Sub

    Private Sub ButtonUpdate_Click()
        Dim sql As String
        sql = "Update TableMarkList Set name='" & TextBox2.Text & "',marks=" &
                                    TextBox3.Text & " Where ID=" & TextBox1.Text
        Dim dbw As New ClassDB()
        dbw.IUD(sql)
        MsgBox("Updated")
    End Sub

    Private Sub ButtonDelete_Click()
        Dim sql As String
        sql = "Delete From TableMarkList Where ID=" & TextBox1.Text
        Dim dbw As New ClassDB()
        dbw.IUD(sql)
        MsgBox("Deleted")
    End Sub

    Private Sub ButtonAll_Click()
        Dim sql As String
        sql = "Select * From TableMarkList"
        Dim dbw As New ClassDB()
        dbw.FillDS(sql)
        DataGridView1.DataSource = dbw.ds.Tables(0)

    End Sub
End Class
```

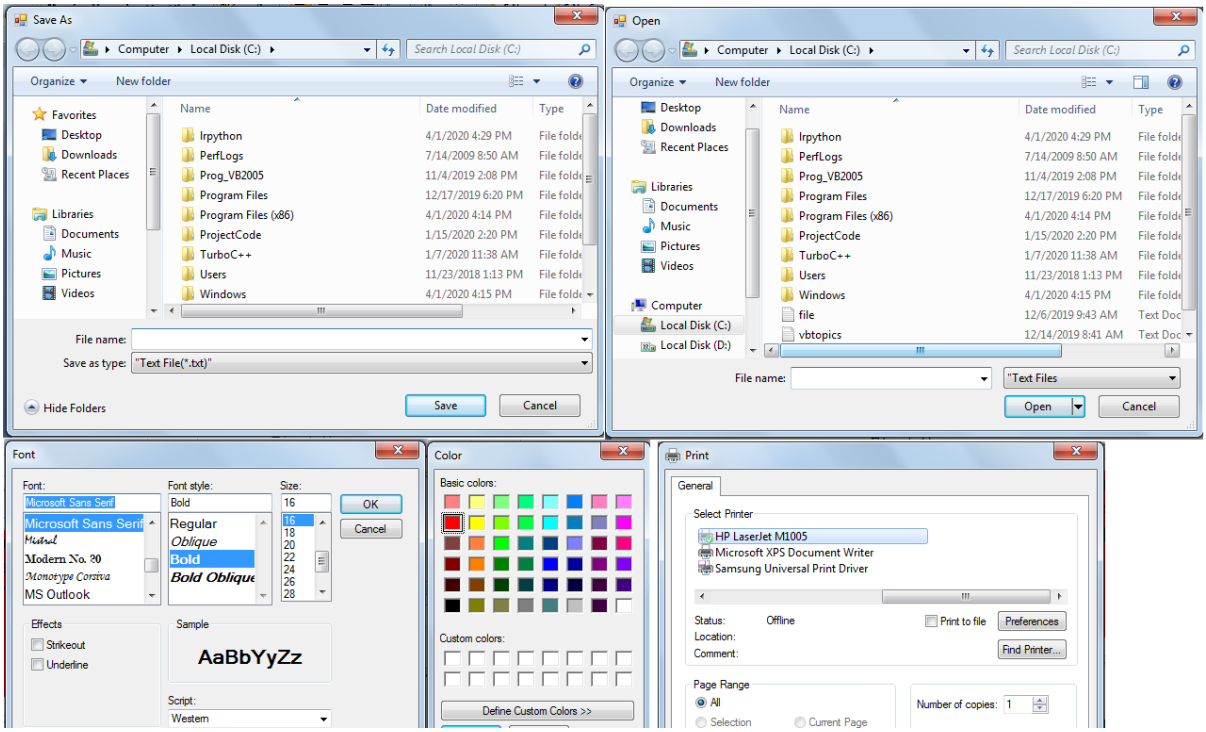
36. Working with crystal report to print records.




```
Imports System.Data.SqlClient
Imports CrystalDecisions.CrystalReports.Engine

Public Class Form04CrystalReport
    Private Sub Button1_Click()
        Dim constr As String
        constr = "Data Source=.\SQLEXPRESS;AttachDbFilename=|DataDirectory|
                          SqlDb.mdf;Integrated Security=True;User Instance=True"
        Dim con As New SqlConnection(constr)
        If con.State <> ConnectionState.Open Then
            con.Open()
        End If
        Dim sql As String
        sql = "Select * From TableMarkList"
        Dim cmd As New SqlCommand(Sql, con)
        Dim da As New SqlDataAdapter(cmd)
        Dim ds As New DataSet()
        da.Fill(ds)

        Dim cr As New ReportDocument
        cr.Load("D:\ComputerEPrograms\VBdotNET\Prog_VB2005\WindowsApplication1\
                                                        CrystalReport1.rpt")
        cr.SetDataSource(ds.Tables(0))
        CrystalReportViewer1.ReportSource = cr
    End Sub
End Class
```

37. Working with different dialogboxes like save dialogbox, open dialog box, font dialog box, color dialog box and print dialog box.






```
Imports System.io
Public Class Form05DialogBox
    Private Sub ButtonSave_Click()
        If SaveFileDialog1.ShowDialog() <> Windows.Forms.DialogResult.Cancel Then
            Dim fw As FileStream = New FileStream(SaveFileDialog1.FileName,
                                        FileMode.Append, FileAccess.Write)
            Dim w As New StreamWriter(fw)
            w.WriteLine(TextBox1.Text)
            w.Flush()
            w.Close()
        End If
    End Sub

    Private Sub ButtonOpen_Click()
        If OpenFileDialog1.ShowDialog() <> Windows.Forms.DialogResult.Cancel Then
            Dim fr As FileStream = New FileStream(OpenFileDialog1.FileName,
                            FileMode.Open, FileAccess.Read)
            Dim r As New StreamReader(fr)
            TextBox1.Text = r.ReadToEnd()
            r.Close()
        End If
    End Sub

    Private Sub ButtonFont_Click()
        If FontDialog1.ShowDialog() <> Windows.Forms.DialogResult.Cancel Then
            TextBox1.Font = FontDialog1.Font
        End If
    End Sub

    Private Sub ButtonColor_Click()
        If ColorDialog1.ShowDialog() <> Windows.Forms.DialogResult.Cancel Then
            TextBox1.BackColor = ColorDialog1.Color
        End If
    End Sub
    Private Sub ButtonPrint_Click()
        PrintDialog1.ShowDialog()
    End Sub


End Class
```