Chapter 9: Lists

Section 9.1: Add items to a List

```
Dim aList as New List(Of Integer)
aList.Add(1)
aList.Add(10)
aList.Add(1001)
```

To add more than one item at a time use **AddRange**. Always adds to the end of the list

```
Dim blist as New List(of Integer)
blist.AddRange(alist)


Dim aList as New List(of String)
alist.AddRange({"one", "two", "three"})
```

In order to add items to the middle of the list use **Insert**

Insert will place the item at the index, and renumber the remaining items

```
Dim aList as New List(Of String)
aList.Add("one")
aList.Add("three")
alist(0) = "one"
alist(1) = "three"
alist.Insert(1,"two")
```

New Output:

```
alist(0) = "one"
alist(1) = "two"
alist(2) = "three"
```

Section 9.2: Check if item exists in a List

```
    Sub Main()
        Dim People = New List(Of String)({"Bob Barker", "Ricky Bobby", "Jeff Bridges"})
        Console.WriteLine(People.Contains("Rick James"))
        Console.WriteLine(People.Contains("Ricky Bobby"))
        Console.WriteLine(People.Contains("Barker"))
        Console.Read
    End Sub
```

Produces the following output:

```
False
True
False
```

Section 9.3: Loop through items in list

```
Dim aList as New List(Of String)
```

```
aList.Add("one")
aList.Add("two")
aList.Add("three")

For Each str As String in aList
    System.Console.WriteLine(str)
Next
```

Produces the following output:

```
one
two
three
```

Another option, would be to loop through using the index of each element:

```
Dim aList as New List(Of String)
aList.Add("one")
aList.Add("two")
aList.Add("three")

For i = 0 to aList.Count - 1 'We use "- 1" because a list uses 0 based indexing.
    System.Console.WriteLine(aList(i))
Next
```

Section 9.4: Create a List

Lists can populated with any data type as necessary, with the format

```
Dim aList as New List(Of Type)
```

For example:

Create a new, empty list of Strings

```
Dim aList As New List(Of String)
```

Create a new list of strings, and populate with some data

VB.NET 2005/2008:

```
Dim aList as New List(Of String)(New String() {"one", "two", "three"})
```

VB.NET 2010:

```
Dim aList as New List(Of String) From {"one", "two", "three"}
```

--

VB.NET 2015:

```
Dim aList as New List(Of String)(New String() {"one", "two", "three"})
```

NOTE:

If you are receiving the following when the code is ran:

> Object reference not set to an instance of an object.

Make sure you either declare as `New` i.e. `Dim aList as New List(Of String)` or if declaring without the `New`, make sure you set the list to a new list - `Dim aList as List(Of String) = New List(Of String)`

Section 9.5: Remove items from a List

```
Dim aList As New List(Of String)
aList.Add("Hello")
aList.Add("Delete Me!")
aList.Add("World")

'Remove the item from the list at index 1
aList.RemoveAt(1)

'Remove a range of items from a list, starting at index 0, for a count of 1)
'This will remove index 0, and 1!
aList.RemoveRange(0, 1)

'Clear the entire list
alist.Clear()
```

Section 9.6: Retrieve items from a List

```
Dim aList as New List(Of String)
aList.Add("Hello, World")
aList.Add("Test")

Dim output As String = aList(0)
```

output:

> Hello, World

If you do not know the index of the item or only know part of the string then use the **Find** or **FindAll** method

```
Dim aList as New List(Of String)
aList.Add("Hello, World")
aList.Add("Test")

Dim output As String = aList.Find(Function(x) x.StartWith("Hello"))
```

output:

> Hello, World

The **FindAll** method returns a new List (of String)

```
Dim aList as New List(Of String)
aList.Add("Hello, Test")
aList.Add("Hello, World")
aList.Add("Test")
```

```vb
Dim output As String = aList.FindAll(Function(x) x.Contains("Test"))
```

> output(0) = "Hello, Test"

> output(1) = "Test"