1. Basic Knowledge

1a. [4 points, one point each] Define/explain any four (only four) of the following terms.

   **Control array:**

   *Group, collection etc. of objects where each object has a unique position within the collection, beginning at location 0 and ending at location N-1 (where N is the total number of elements within the group).*

   **Loop index:**

   *Counter that keeps track of the iterations within a loop. After each iteration, loop index is incremented (or decremented) by 1 (default) or by some other user-specified amount.*

   **Divide and conquer:**

   “Breaking up” a large problem into smaller sub-problems, solving the smaller sub-problems and then combining the results of these smaller sub-problems such that the solution to the original problem is obtained.

   **Procedure:**

   *A sub-program that does not return a value to the caller.*

   **“Pass” by value:**

   *Sending a copy of an argument to a function/procedure as opposed to the actual argument (variable representing the argument) itself. Any changes made to the argument within the sub-program are local to the sub-program (e.g., the original variable is not affected).*
1b. [10 points, 1 point each] **True / False.** For each question, circle your choice of either True or False (do not circle both!).

1) In addition to properties, an object can also have **procedures/methods** associated with it.  
   **True**  **False**

2) A **control array** must always include a Frame  
   **True**  **False**

3) A **counted loop** can always be written as a **conditional loop**  
   **True**  **False**

4) In a group of CheckBoxes, only one can be selected  
   **True**  **False**

5) The following loop will **iterate** 50 times  
   ```vbnet
   final = 10
   increase = 1
   For index = 1 To final Step increase
       final = 50
   Next
   ```  
   **True**  **False**

6) The result of the following Boolean expression will be **True**  
   ```vbnet
   ("50" < "100")
   ```  
   **True**  **False**

7) Given a control array of option controls (called **optArray**),  
   the following statement will set the **Value** property of the first option control of **optArray** to False  
   ```vbnet
   optArray(1).value = False
   ```  
   **True**  **False**

8) An event handler is an example of a **function** subprogram  
   **True**  **False**

9) A **procedure** subprogram cannot return a value  
   **True**  **False**

10) A **TextBox** object can only support vertical scroll bars  
    **True**  **False**
1c. [3 points] List three **advantages** of modularisation.

- Promotes re-use of code
- Divide and conquer
- Makes code easier to read/follow
- Makes code easier to maintain and update
- etc etc etc.

1d. [2 points] Describe/explain how we can set up a control array of Option controls (objects) using the **Frame method**.

- Place a Frame object on the Form
- Place a control object (e.g., Option control) within the Frame
- Copy the control object that is in the Frame
- Paste the control object
- When Visual basic prompts you whether you want to create a control array, click “Yes”
- Paste any additional controls within the array as required

1e. [2 points] Describe/explain the differences between a **counted loop** and a **conditional loop**.

*Counted Loop: loop iterates a pre-specified number of times – typically used when we know how many times loops should iterate*

*Conditional loop: Loop iterates based on the value of some condition. Iterate provided the condition is True.*
2. Programming

2a. [3 points] Provide the definition of a procedure (or sub) subprogram called myProcedure that takes two arguments: the first argument, called arg1 is of type Integer and passed by reference and the second argument, called arg2 is of type Double and passed by value.

Note: You do not need to provide any statements for the body of this procedure.

    Private Sub myProcedure(ByRef arg1 As Integer, ByVal arg2 As Double)
     End Sub

2b. [2 points] Assume you have a control array called myOptionArray that contains three Option controls. Assume that you also have a Button control called btnControl. Is the following code segment valid? Explain your answer.

    Private Sub btnControl_Click()
       myOptionArray(3).Value = True
    End Sub

   No, this is invalid. In a control array, the index (indicating the position of an element within the array) ranges from 0 – N-1 (where N is the total number of elements within the control array). Therefore, since we have three Option controls within this control array, the index will range from 0 – 2 only.

2c. [1 point] Assume txtResult is a TextBox placed on a Form. List the output displayed in the txtResult TextBox after the following code segment is executed.

    Dim myString As String
    Dim findString As String
    myString = “Practice, practice makes perfect”
    findString = “pra”
    txtResult.Text = Mid(myString, 10, 5)

   Output: txtResult.Text = “_prac” (where “_” denotes a space)
2d. [1 point] Assume `txtResult` is a TextBox placed on a Form. List the output displayed in the `txtResult` TextBox after the following code segment is executed.

```vba
Dim myString As String
Dim findString As String

myString = “Practice, practice makes perfect”
findString = “pra”
txtResult.Text = CStr(InStr(myString, findString))
```

**Output:** `txtResult.Text = 11`

2e. [1 point] Assume `txtResult` is a TextBox placed on Form. List the output displayed in the `txtResult` TextBox after the following code segment is executed.

```vba
Dim myStr As String
Dim findStr As String
Dim replaceStr As String

myString = “Practice, practice makes perfect”
findString = “cti”
replaceString = “itc”
txtResult.Text = Replace(myStr, findStr, replaceStr, InStr(8, myString, findString), 1)
```

**Output:** `txtResult.Text = “itcce makes perfect”`
2f. [4 points] Write a function subprogram called `computeSum` that takes two arguments, a **Single** (called `myNumber`) and an **Integer** (called `num`) that will compute and return to the user the following value: 

\[(\text{myNumber} \times n)\]. You must use a loop to compute the value to obtain full marks – in other words, you can’t simply multiply \textbf{myNumber} by \textbf{n}.

\textbf{Hint:} Think of what it means to multiply a number by \textbf{n} (e.g., addition).

```vbnet
Private Function computeSum(myNumber As Single, n As Integer) As Single
    Dim sum As Single
    Dim loopIndex As Integer
    sum = 0
    For loopIndex = 1 To n
        sum = sum + myNumber
    Next
    computeSum = sum
End Function
```

2f. [2 point] Use the function you wrote in part b to complete the following code segment.

```vbnet
Dim value As Single
Dim num As Integer

value = 2.0
num = 10

' Declare the appropriate variable (called sum) to assign the ' return value of the function computeSum.

Dim sum As Single
sum = computeSum(value, num)
```
2g. [5 points] Write a procedure (or sub) subprogram called `reverseStr` that takes a single String argument (called `inputStr` that is passed by reference) and performs the following operations: i) displays the characters of `inputStr` in a ListBox called `list1` (one character per line) in reverse order and ii) displays the reverse of `inputStr` in a TextBox called `text1`. For example, given the String “abcd”, the TextBox output will be “dcba” and the ListBox output will be:

d
c
b
a

**Note:** You cannot make use of Visual Basic’s `ReverseStr` function!

```vbnet
Private Sub reverseStr(ByRef inputStr As String)
    Dim char As String
    Dim result As String
    Dim loopIndex As Integer

    For loopIndex = Len(inputStr) To 1 Step -1
        char = Mid(inputStr, loopIndex, 1)
        result = result + char
        list1.AddItem(char)
    Next
    text1.text = result
End Sub
```
Additional Space

Use this page for any additional space you require. Please state question numbers.
String-Related Functions

**String:**

- **Asc(String) As Integer** - Returns the character code for the first character in the string
- **Chr(Long) As String** - Returns the character string corresponding to the specified code
- **InStr([Start As Integer], String1, String2) As Long** - specifying the position of the first occurrence of String2 in String1, starting at Start if the argument is specified, otherwise at the beginning of String1
- **InStrRev(String1, String2, [Start As Integer]) As Long** - specifying the position of the first occurrence of String2 in String1, from the end of String1 (or from Start if the argument is specified)
- **LCase(String1) As String** - Returns String1 converted to lower case
- **Left(String1, Integer) As String** - Returns a string containing the specified number of characters from the left of String1
- **Len(String1) As Long** - Returns a Long, the number of characters in String1
- **Ltrim(String1) As String** - Returns a string with blanks removed from the left
- **Mid(String, Start(Long), [Length As Long]) As String** - Returns all (or Length if it is specified) characters from a string starting at position Start
- **Replace(String1, String2, String3, [Start], [Count]) As String** - Returns a string with String2 replaced by String3, wherever it is found in String1, beginning at position Start, or replaces it Count times from position Start
- **Right(String1, Integer) As String** - Returns a string containing the specified number of characters from the right of String1
- **Rtrim(String1) As String** - Returns a string with blanks removed from the right
- **Space(Long) As String** - Returns a string composed of just blanks, as many as specified by Long
- **StrComp(String1, String2) As Integer** - Returns an integer indicating the comparison of String1 and String2, namely -1, 0 or +1 depending if String1 is less than, equal to, or greater than String2
- **String(Long, String1) As String** - Returns a string composed of just the first character of String1, as many as specified by Long
- **StrReverse(String1) As String** - Returns a string composed of the characters from String1 but in reverse order
- **UCase(String1) As String** - Returns String1 converted to uppercase