Question 1

1.

```
Factorial(n)
        if n == 0
                Return 1
        Else
                Return n *Factorial(n-1)
        End if
End Function
```

2.

```
Search(Array, Element, Start, End)
        If End > Start
                Mid = Start + (End – Start)/2
                If Element == Array[Mid]
                        Return Mid
                End if
                If Array[Mid] > Element
                        Return Search(Array, Element, Start, Mid -1)
                Else Array[Mid] < Element
                        Return Search(Array, Element, Mid +1,End)
                End if
        Return -1
End Function
```

3.

```
Palindrome(String, Start, End)
    If Start >= End
        Return True
    End if
    If String[Start] == String[End]
        Return Palindrome(String, Start+1, End-1)
    Else
        Return False
    End if
End Function
```

Question 2

1.O(n)

2.O(logn)

3.O(n)

Question 3

The Merge algorithm is used to combine two sorted arrays into a single sorted array. In the worst case we're going through each element in both arrays exactly once. So, the time complexity of the Merge operation is linear with respect to the total number of elements in the two arrays.

Therefore, the worst-case time complexity of the Merge algorithm is O(n).