**Problem Statement 1:**

**Merge Sort:** **Code:**

**import** org.apache.spark.{SparkConf, SparkContext}  
  
**object** Mergesort {  
 **def** main(args: Array[String]) {  
 System.*setProperty*("hadoop.home.dir","C:\\Users\\nikit\\OneDrive\\Desktop\\winutils" )  
 **val** conf = **new** SparkConf().setAppName("Breadthfirst").setMaster("local[\*]")  
 **val** sc = **new** SparkContext(conf)  
 **val** list = *List*(10,7,5,8,4,3,1,2)  
 **def** mergeSort(a: List[Int]): List[Int] = {  
 **val** n = a.length / 2  
 **if** (n == 0) a  
 **else** {  
 **def** merge(a: List[Int], b: List[Int]): List[Int] =  
 (a, b) **match** {  
 **case**(*Nil*, b) => b  
 **case**(a, *Nil*) => a  
 **case**(x *::* as, y *::* bs) =>  
 **if** (x < y) x::merge(as, b)  
 **else** y :: merge(a, bs)  
 }  
 **val** (left, right) = a splitAt(n)  
 merge(mergeSort(left), mergeSort(right))  
 }  
 }  
 **val** result=mergeSort(list)  
 *print*(result)  
  
 }  
}

**Output:**

https://github.com/NikithaPateel/Big_Data_Programming/raw/master/Module2_ICP2/screenshots/output1.JPG

Problem Statement 2:

**Breadth First Search:** **Code:**

**import** org.apache.spark.{SparkConf, SparkContext}  
  
**object** BreadthFirstSearch {  
 **def** main(args: Array[String]) {  
 System.*setProperty*("hadoop.home.dir","C:\\Users\\nikit\\OneDrive\\Desktop\\winutils" )  
 **val** conf = **new** SparkConf().setAppName("Breadthfirst").setMaster("local[\*]")  
 **val** sc = **new** SparkContext(conf)  
 **type** Vertex = Int  
 **type** Graph = Map[Vertex, List[Vertex]]  
 **val** g: Graph = *Map*(1 -> *List*(2,3,5,6,7), 2 -> *List*(1,3,4,6,7), 3 -> *List*(1,2), 4 -> *List*(2,5,7),5 -> *List*(1,6,7),6 -> *List*(1,2,5,7),7 -> *List*(1,2,4,5,6))  
  
 **def** BFS(start: Vertex, g: Graph): List[List[Vertex]] = {  
  
 **def** BFS0(elems: List[Vertex],visited: List[List[Vertex]]): List[List[Vertex]] = {  
 **val** newNeighbors = elems.flatMap(g(\_)).filterNot(visited.flatten.contains).distinct  
 **if** (newNeighbors.isEmpty)  
 visited  
 **else** BFS0(newNeighbors, newNeighbors :: visited)  
 }  
 BFS0(*List*(start),*List*(*List*(start))).reverse  
 }  
 **val** result=BFS(1,g )  
 *println*(result.mkString(","))  
  
 }  
  
  
}

**Output:**

https://github.com/NikithaPateel/Big_Data_Programming/raw/master/Module2_ICP2/screenshots/output2.JPG