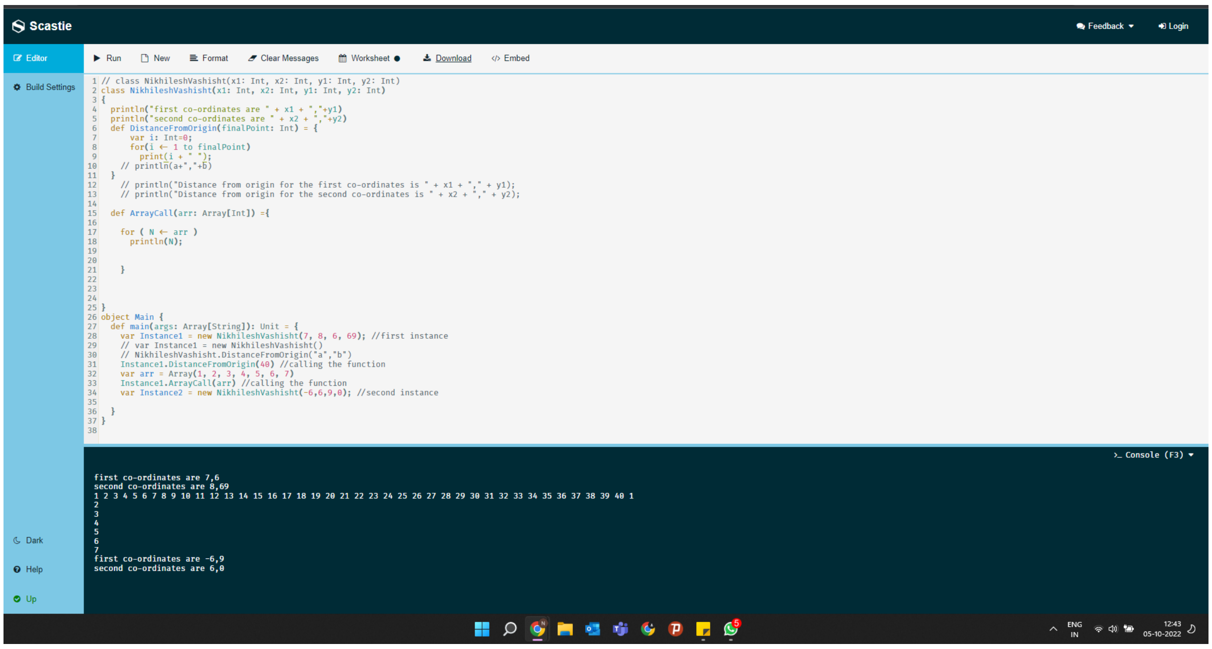
1) Explain what is the reason for the upper limit of 100X improvement in speed often cited for Apache Spark? In what cases do you see such improvements?
    a) In the case of Hadoop Map Reduce , data is written to HDFS three times throughout each step (map, shuffle, and reduce). Spark doesn't save anything unless an action is taken on the data. Even if we apply an action, such as saveAsTextFile, each executor will conduct the action in parallel and write data to the file, speeding up spark even more. Consider doing hundreds of map operations in Map Reduceand Spark. Spark maintains an executor JVM running on every node, so starting a job is as simple as making an RPC to it and giving a Runnable to a thread pool, which takes just milliseconds.
    b) There are several more factors, including the serialisation of data in Map Reduce and Spark, resource usage, Spark's on-the-fly partitioning, and its most crucial component, the tungsten catalyst. Instead of using many stages, Spark utilises DAG to complete all necessary optimization and processing in a single step. Even without users invoking persist, Spark automatically persists certain intermediate data in shuffle operations. In the event that a node fails during the shuffle, this prevents the need to recompute the entire input.
    c) Undeniably Spark is quick but we can still make sure to improve the performance of Spark by tuning it. Shuffle, which in theory involves physically moving data across a network and writing it to disc, incurs costs due to network, disc, and data serialisation. When the data has already been partitioned using the same partitioner by a previous transformation, Spark is aware to avoid doing a shuffle.
       Bucketing which is it distributes data by hashing the bucket value among a given number of buckets.

2) Using Scala, write a class of your choice (not the Employee class that was discussed in class). Include your name in this class so it is identifiable. Example: CarsSaber The class must have at least two (2) fields and two methods/functions. Create two instances (objects) of this class in the main. Using the object created, print the fields and call the method/function at least once. Provide the code (textfile) and a screenshot of your output with the code (on Scastie).

Screenshot:



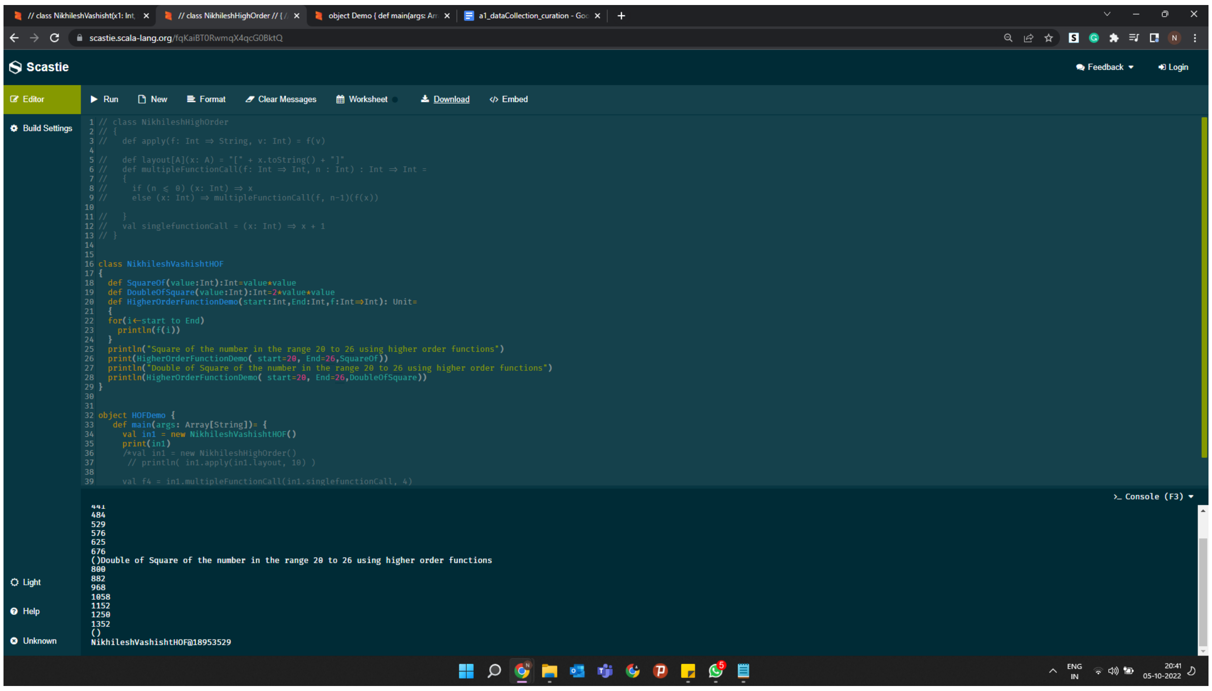
```scala
// class NikhileshVashisht(x1: Int, x2: Int, y1: Int, y2: Int)
class NikhileshVashisht(x1: Int, x2: Int, y1: Int, y2: Int)
{
  println("first co-ordinates are " + x1 + ","+y1)
  println("second co-ordinates are " + x2 + ","+y2)
  def DistanceFromOrigin(finalPoint: Int) = {
      var i: Int=0;
      for(i ← 1 to finalPoint)
        print(i + " ");
    // println(a+","+b)
  }
    // println("Distance from origin for the first co-ordinates is " + x1 + "," + y1);
    // println("Distance from origin for the second co-ordinates is " + x2 + "," + y2);

  def ArrayCall(arr: Array[Int]) ={

    for ( N ← arr )
      println(N);


    }




}
object Main {
  def main(args: Array[String]): Unit = {
    var Instance1 = new NikhileshVashisht(7, 8, 6, 69); //first instance
    // var Instance1 = new NikhileshVashisht()
    // NikhileshVashisht.DistanceFromOrigin("a","b")
    Instance1.DistanceFromOrigin(40) //calling the function
    var arr = Array(1, 2, 3, 4, 5, 6, 7)
    Instance1.ArrayCall(arr) //calling the function
    var Instance2 = new NikhileshVashisht(-6,6,9,0); //second instance

  }
}

```

```
first co-ordinates are 7,6
second co-ordinates are 8,69
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 1
2
3
4
5
6
7
first co-ordinates are -6,9
second co-ordinates are 6,0
```

3) Create another class and write at least 2 functions where at least one function is a higher order function – ie this function takes a function as a parameter. Your higher order function should do something useful as demonstrated in the class example. Just a higher order function that take in a function as a parameter but does nothing the input or does something trivial.

Explanation: a higher order function is a function which either takes a function as an argument or returns a function as a result. In my code below I have taken 2 functions **SquareOf** and **DoubleOfSquare** as a parameter to the third function **HigherOrderFunctionDemo**

Screenshot:



```scala
// class NikhileshHighOrder
// {
//   def apply(f: Int ⇒ String, v: Int) = f(v)

//   def layout[A](x: A) = "[" + x.toString() + "]"
//   def multipleFunctionCall(f: Int ⇒ Int, n : Int) : Int ⇒ Int =
//   {
//     if (n ≤ 0) (x: Int) ⇒ x
//     else (x: Int) ⇒ multipleFunctionCall(f, n-1)(f(x))

//   }
//   val singlefunctionCall = (x: Int) ⇒ x + 1
// }


class NikhileshVashishtHOF
{
  def SquareOf(value:Int):Int=value*value
  def DoubleOfSquare(value:Int):Int=2*value*value
  def HigherOrderFunctionDemo(start:Int,End:Int,f:Int⇒Int): Unit=
  {
  for(i←start to End)
    println(f(i))
  }
  println("Square of the number in the range 20 to 26 using higher order functions")
  print(HigherOrderFunctionDemo( start=20, End=26,SquareOf))
  println("Double of Square of the number in the range 20 to 26 using higher order functions")
  println(HigherOrderFunctionDemo( start=20, End=26,DoubleOfSquare))
}


object HOFDemo {
   def main(args: Array[String])= {
     val in1 = new NikhileshVashishtHOF()
     print(in1)
     /*val in1 = new NikhileshHighOrder()
      // println( in1.apply(in1.layout, 10) )

     val f4 = in1.multipleFunctionCall(in1.singlefunctionCall, 4)
```

```
441
484
529
576
625
676
()Double of Square of the number in the range 20 to 26 using higher order functions
800
882
968
1058
1152
1250
1352
()
NikhileshVashishtHOF@18953529
```

4) Explain in you own word why higher order functions are more efficient in a Big Data setting than just having the function as a method within an class?

a) Higher order functions enable us to modify a function's behaviour by adding new behaviour to the one it already has. Higher order functions is made for reuasbility.

b) Higher order functions gives us a extract over the control flow which is very useful in Big Data setting as it gives the user more control over the function handling. Suppose we are using complex data structures such as maps or structs or arrays, we can process these complex data structures better using higher order functions.

c) Higher order function removes redundancy in the code which makes it easier to read and understand the code. User defined higher order functions will help in reducing the duplicates while processing in big data environment which can reduce the computation time.