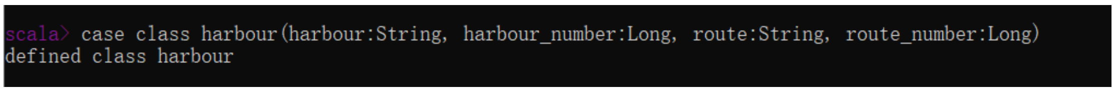
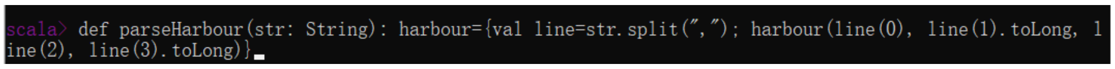
Q1



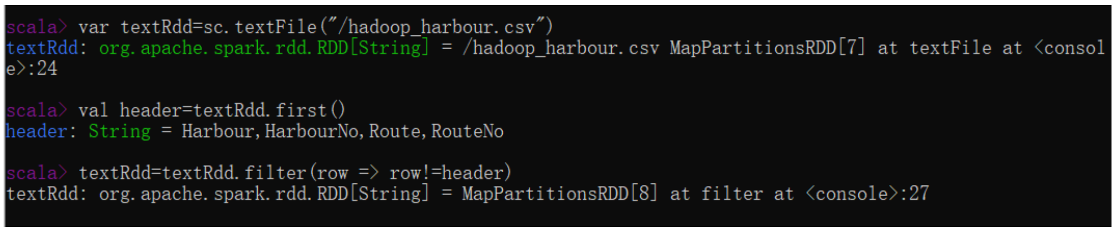
```
scala> case class harbour(harbour:String, harbour_number:Long, route:String, route_number:Long)
defined class harbour
```

case class harbour(harbour:String, harbour_number:Long, route:String, route_number:Long)



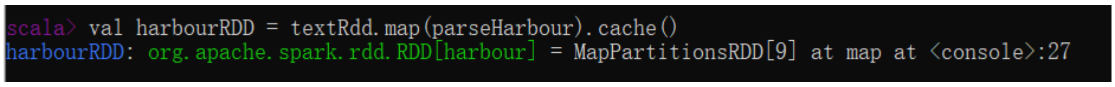
```
scala> def parseHarbour(str: String): harbour={val line=str.split(","); harbour(line(0), line(1).toLong, l
ine(2), line(3).toLong)}
```

def parseHarbour(str: String): harbour={val line=str.split(","); harbour(line(0), line(1).toLong, line(2), line(3).toLong)}



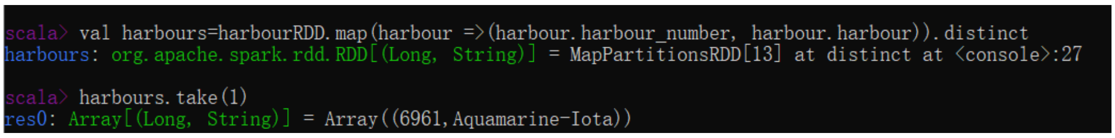
```
scala> var textRdd=sc.textFile("/hadoop_harbour.csv")
textRdd: org.apache.spark.rdd.RDD[String] = /hadoop_harbour.csv MapPartitionsRDD[7] at textFile at <consol
e>:24

scala> val header=textRdd.first()
header: String = Harbour,HarbourNo,Route,RouteNo

scala> textRdd=textRdd.filter(row => row!=header)
textRdd: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[8] at filter at <console>:27
```

var textRdd=sc.textFile("/hadoop_harbour.csv")
val header=textRdd.first()
textRdd=textRdd.filter(row => row!=header)



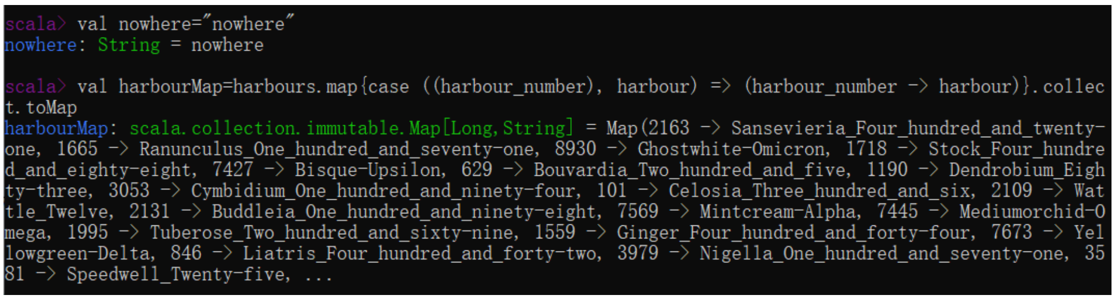
```
scala> val harbourRDD = textRdd.map(parseHarbour).cache()
harbourRDD: org.apache.spark.rdd.RDD[harbour] = MapPartitionsRDD[9] at map at <console>:27
```

val harbourRDD = textRdd.map(parseHarbour).cache()



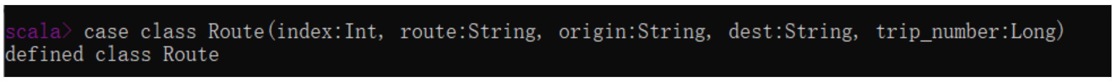
```
scala> val harbours=harbourRDD.map(harbour =>(harbour.harbour_number, harbour.harbour)).distinct
harbours: org.apache.spark.rdd.RDD[(Long, String)] = MapPartitionsRDD[13] at distinct at <console>:27

scala> harbours.take(1)
res0: Array[(Long, String)] = Array((6961,Aquamarine-Iota))
```

val harbours=harbourRDD.map(harbour =>(harbour.harbour_number, harbour.harbour)).distinct
harbours.take(1)



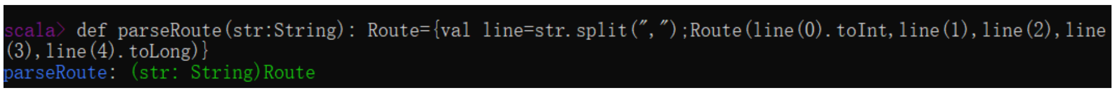
```
scala> val nowhere="nowhere"
nowhere: String = nowhere

scala> val harbourMap=harbours.map{case ((harbour_number), harbour) => (harbour_number -> harbour)}.collec
t.toMap
harbourMap: scala.collection.immutable.Map[Long,String] = Map(2163 -> Sansevieria_Four_hundred_and_twenty-
one, 1665 -> Ranunculus_One_hundred_and_seventy-one, 8930 -> Ghostwhite-Omicron, 1718 -> Stock_Four_hundre
d_and_eighty-eight, 7427 -> Bisque-Upsilon, 629 -> Bouvardia_Two_hundred_and_five, 1190 -> Dendrobium_Eigh
ty-three, 3053 -> Cymbidium_One_hundred_and_ninety-four, 101 -> Celosia_Three_hundred_and_six, 2109 -> Wat
tle_Twelve, 2131 -> Buddleia_One_hundred_and_ninety-eight, 7569 -> Mintcream-Alpha, 7445 -> Mediumorchid-O
mega, 1995 -> Tuberose_Two_hundred_and_sixty-nine, 1559 -> Ginger_Four_hundred_and_forty-four, 7673 -> Yel
lowgreen-Delta, 846 -> Liatris_Four_hundred_and_forty-two, 3979 -> Nigella_One_hundred_and_seventy-one, 35
81 -> Speedwell_Twenty-five, ...
```

val nowhere="nowhere"
val harbourMap=harbours.map{case ((harbour_number), harbour) => (harbour_number -> harbour)}.collect.toMap



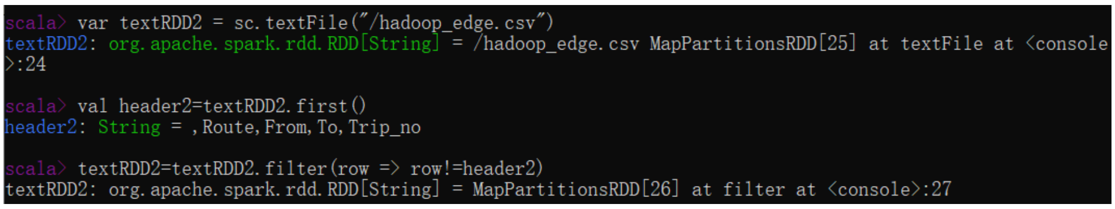
```
scala> case class Route(index:Int, route:String, origin:String, dest:String, trip_number:Long)
defined class Route
```

case class Route(index:Int, route:String, origin:String, dest:String, trip_number:Long)



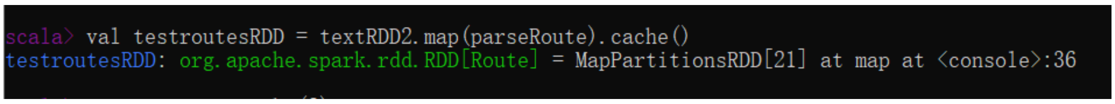
```
scala> def parseRoute(str:String): Route={val line=str.split(",");Route(line(0).toInt,line(1),line(2),line
(3),line(4).toLong)}
parseRoute: (str: String)Route
```

def parseRoute(str:String): Route={val line=str.split(",");Route(line(0).toInt,line(1),line(2),line (3),line(4).toLong)}

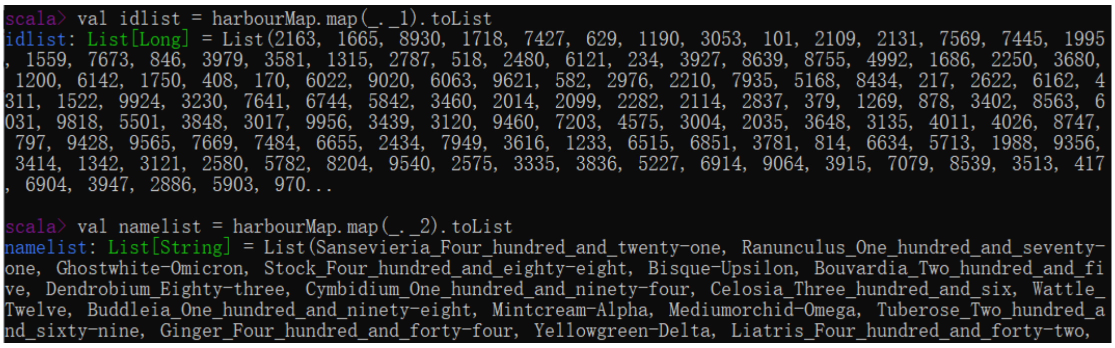


```
scala> var textRDD2 = sc.textFile("/hadoop_edge.csv")
textRDD2: org.apache.spark.rdd.RDD[String] = /hadoop_edge.csv MapPartitionsRDD[25] at textFile at <console
>:24

scala> val header2=textRDD2.first()
header2: String = ,Route,From,To,Trip_no

scala> textRDD2=textRDD2.filter(row => row!=header2)
textRDD2: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[26] at filter at <console>:27
```

var textRDD2 = sc.textFile("/hadoop_edge.csv")
val header2=textRDD2.first()
textRDD2=textRDD2.filter(row => row!=header2)

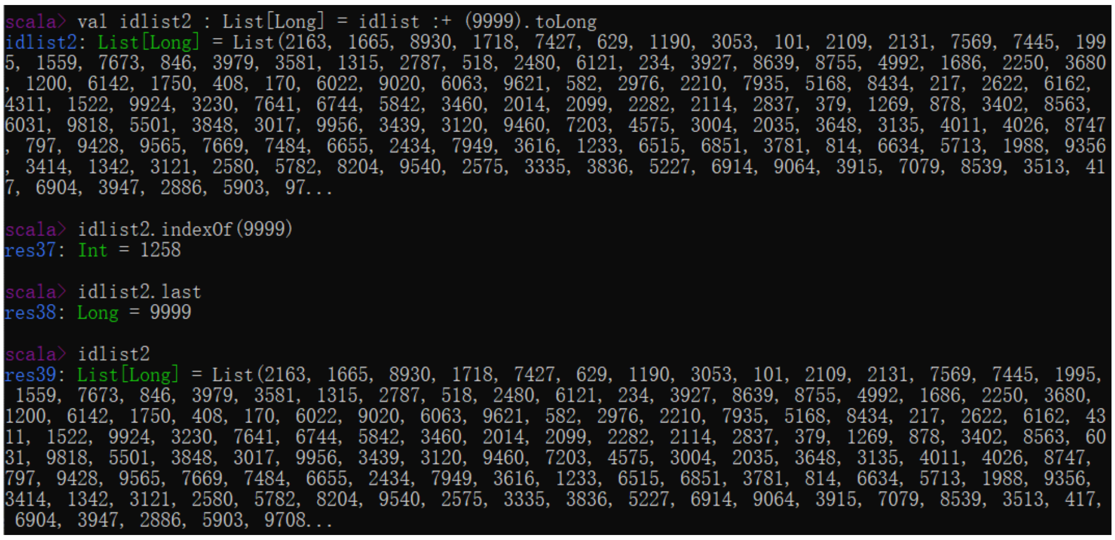


```
scala> val testroutesRDD = textRDD2.map(parseRoute).cache()
testroutesRDD: org.apache.spark.rdd.RDD[Route] = MapPartitionsRDD[21] at map at <console>:36
```



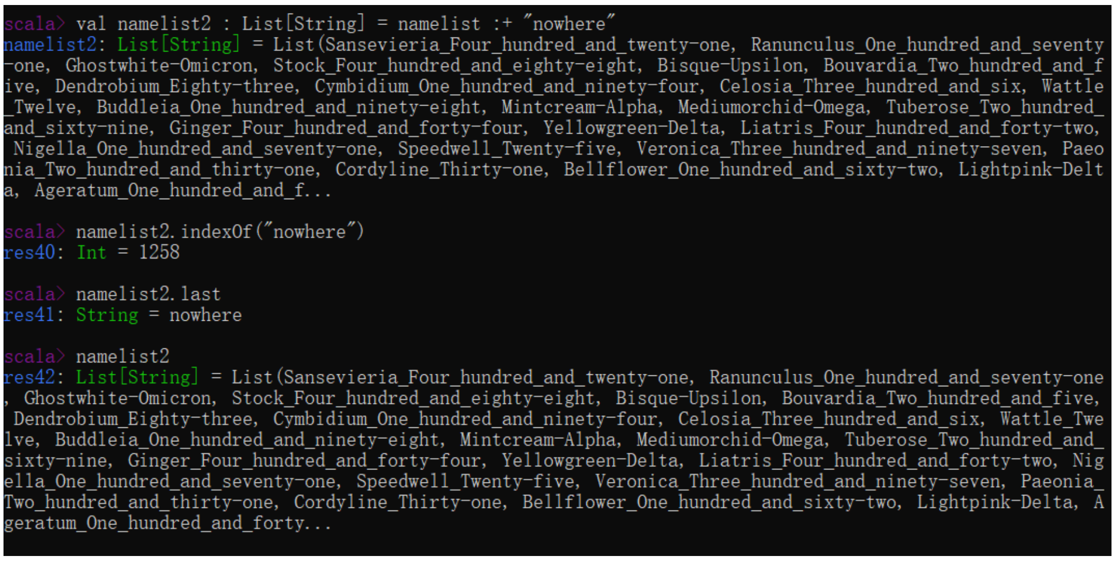
```
scala> val idlist = harbourMap.map(_._1).toList
idlist: List[Long] = List(2163, 1665, 8930, 1718, 7427, 629, 1190, 3053, 101, 2109, 2131, 7569, 7445, 1995
, 1559, 7673, 846, 3979, 3581, 1315, 2787, 518, 2480, 6121, 234, 3927, 8639, 8755, 4992, 1686, 2250, 3680,
 1200, 6142, 1750, 408, 170, 6022, 9020, 6063, 9621, 582, 2976, 2210, 7935, 5168, 8434, 217, 2622, 6162, 4
311, 1522, 9924, 3230, 7641, 6744, 5842, 3460, 2014, 2099, 2282, 2114, 2837, 379, 1269, 878, 3402, 8563, 6
031, 9818, 5501, 3848, 3017, 9956, 3439, 3120, 9460, 7203, 4575, 3004, 2035, 3648, 3135, 4011, 4026, 8747,
 797, 9428, 9565, 7669, 7484, 6655, 2434, 7949, 3616, 1233, 6515, 6851, 3781, 814, 6634, 5713, 1988, 9356,
 3414, 1342, 3121, 2580, 5782, 8204, 9540, 2575, 3335, 3836, 5227, 6914, 9064, 3915, 7079, 8539, 3513, 417
, 6904, 3947, 2886, 5903, 970...

scala> val namelist = harbourMap.map(_._2).toList
namelist: List[String] = List(Sansevieria_Four_hundred_and_twenty-one, Ranunculus_One_hundred_and_seventy-
one, Ghostwhite-Omicron, Stock_Four_hundred_and_eighty-eight, Bisque-Upsilon, Bouvardia_Two_hundred_and_fi
ve, Dendrobium_Eighty-three, Cymbidium_One_hundred_and_ninety-four, Celosia_Three_hundred_and_six, Wattle_
Twelve, Buddleia_One_hundred_and_ninety-eight, Mintcream-Alpha, Mediumorchid-Omega, Tuberose_Two_hundred_a
nd_sixty-nine, Ginger_Four_hundred_and_forty-four, Yellowgreen-Delta, Liatris_Four_hundred_and_forty-two,
```

val testroutesRDD = textRDD2.map(parseRoute).cache()
val idlist = harbourMap.map(_._1).toList
val namelist = harbourMap.map(_._2).toList



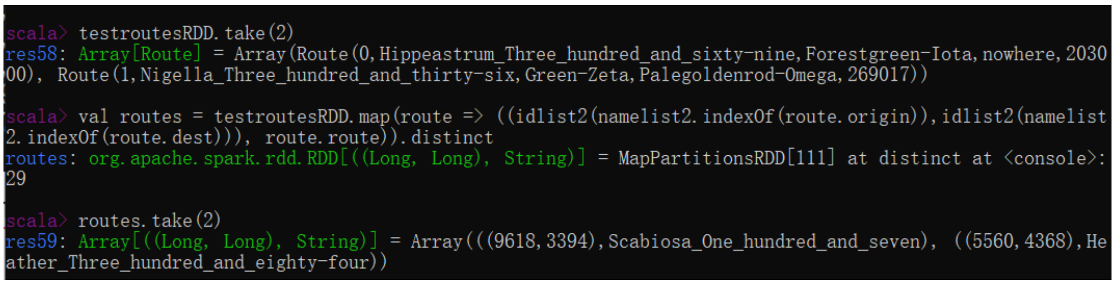
```
scala> val idlist2 : List[Long] = idlist :+ (9999).toLong
idlist2: List[Long] = List(2163, 1665, 8930, 1718, 7427, 629, 1190, 3053, 101, 2109, 2131, 7569, 7445, 199
5, 1559, 7673, 846, 3979, 3581, 1315, 2787, 518, 2480, 6121, 234, 3927, 8639, 8755, 4992, 1686, 2250, 3680
, 1200, 6142, 1750, 408, 170, 6022, 9020, 6063, 9621, 582, 2976, 2210, 7935, 5168, 8434, 217, 2622, 6162,
4311, 1522, 9924, 3230, 7641, 6744, 5842, 3460, 2014, 2099, 2282, 2114, 2837, 379, 1269, 878, 3402, 8563,
6031, 9818, 5501, 3848, 3017, 9956, 3439, 3120, 9460, 7203, 4575, 3004, 2035, 3648, 3135, 4011, 4026, 8747
, 797, 9428, 9565, 7669, 7484, 6655, 2434, 7949, 3616, 1233, 6515, 6851, 3781, 814, 6634, 5713, 1988, 9356
, 3414, 1342, 3121, 2580, 5782, 8204, 9540, 2575, 3335, 3836, 5227, 6914, 9064, 3915, 7079, 8539, 3513, 41
7, 6904, 3947, 2886, 5903, 97...

scala> idlist2.indexOf(9999)
res37: Int = 1258

scala> idlist2.last
res38: Long = 9999

scala> idlist2
res39: List[Long] = List(2163, 1665, 8930, 1718, 7427, 629, 1190, 3053, 101, 2109, 2131, 7569, 7445, 1995,
 1559, 7673, 846, 3979, 3581, 1315, 2787, 518, 2480, 6121, 234, 3927, 8639, 8755, 4992, 1686, 2250, 3680,
1200, 6142, 1750, 408, 170, 6022, 9020, 6063, 9621, 582, 2976, 2210, 7935, 5168, 8434, 217, 2622, 6162, 43
11, 1522, 9924, 3230, 7641, 6744, 5842, 3460, 2014, 2099, 2282, 2114, 2837, 379, 1269, 878, 3402, 8563, 60
31, 9818, 5501, 3848, 3017, 9956, 3439, 3120, 9460, 7203, 4575, 3004, 2035, 3648, 3135, 4011, 4026, 8747,
797, 9428, 9565, 7669, 7484, 6655, 2434, 7949, 3616, 1233, 6515, 6851, 3781, 814, 6634, 5713, 1988, 9356,
3414, 1342, 3121, 2580, 5782, 8204, 9540, 2575, 3335, 3836, 5227, 6914, 9064, 3915, 7079, 8539, 3513, 417,
 6904, 3947, 2886, 5903, 9708...
```

```
val idlist2 : List[Long] = idlist :+ (9999).toLong
idlist2.indexOf(9999)
idlist2.last
idlist2
```



```
scala> val namelist2 : List[String] = namelist :+ "nowhere"
namelist2: List[String] = List(Sansevieria_Four_hundred_and_twenty-one, Ranunculus_One_hundred_and_seventy
-one, Ghostwhite-Omicron, Stock_Four_hundred_and_eighty-eight, Bisque-Upsilon, Bouvardia_Two_hundred_and_f
ive, Dendrobium_Eighty-three, Cymbidium_One_hundred_and_ninety-four, Celosia_Three_hundred_and_six, Wattle
_Twelve, Buddleia_One_hundred_and_ninety-eight, Mintcream-Alpha, Mediumorchid-Omega, Tuberose_Two_hundred_
and_sixty-nine, Ginger_Four_hundred_and_forty-four, Yellowgreen-Delta, Liatris_Four_hundred_and_forty-two,
 Nigella_One_hundred_and_seventy-one, Speedwell_Twenty-five, Veronica_Three_hundred_and_ninety-seven, Paeo
nia_Two_hundred_and_thirty-one, Cordyline_Thirty-one, Bellflower_One_hundred_and_sixty-two, Lightpink-Delt
a, Ageratum_One_hundred_and_f...

scala> namelist2.indexOf("nowhere")
res40: Int = 1258

scala> namelist2.last
res41: String = nowhere

scala> namelist2
res42: List[String] = List(Sansevieria_Four_hundred_and_twenty-one, Ranunculus_One_hundred_and_seventy-one
, Ghostwhite-Omicron, Stock_Four_hundred_and_eighty-eight, Bisque-Upsilon, Bouvardia_Two_hundred_and_five,
 Dendrobium_Eighty-three, Cymbidium_One_hundred_and_ninety-four, Celosia_Three_hundred_and_six, Wattle_Twe
lve, Buddleia_One_hundred_and_ninety-eight, Mintcream-Alpha, Mediumorchid-Omega, Tuberose_Two_hundred_and_
sixty-nine, Ginger_Four_hundred_and_forty-four, Yellowgreen-Delta, Liatris_Four_hundred_and_forty-two, Nig
ella_One_hundred_and_seventy-one, Speedwell_Twenty-five, Veronica_Three_hundred_and_ninety-seven, Paeonia_
Two_hundred_and_thirty-one, Cordyline_Thirty-one, Bellflower_One_hundred_and_sixty-two, Lightpink-Delta, A
geratum_One_hundred_and_forty...
```

```
val namelist2 : List[String] = namelist :+ "nowhere"
namelist2.indexOf("nowhere")
namelist2.last
namelist2
```

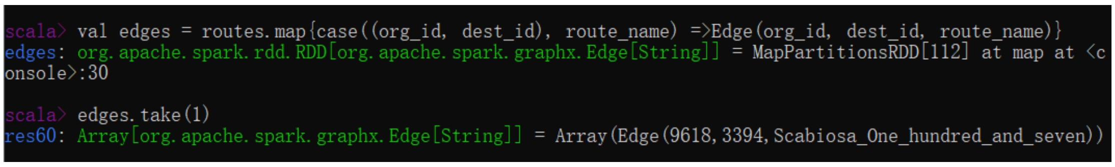


```
scala> testroutesRDD.take(2)
res58: Array[Route] = Array(Route(0,Hippeastrum_Three_hundred_and_sixty-nine,Forestgreen-Iota,nowhere,2030
00), Route(1,Nigella_Three_hundred_and_thirty-six,Green-Zeta,Palegoldenrod-Omega,269017))

scala> val routes = testroutesRDD.map(route => ((idlist2(namelist2.indexOf(route.origin)),idlist2(namelist
2.indexOf(route.dest))), route.route)).distinct
routes: org.apache.spark.rdd.RDD[((Long, Long), String)] = MapPartitionsRDD[111] at distinct at <console>:
29

scala> routes.take(2)
res59: Array[((Long, Long), String)] = Array(((9618,3394),Scabiosa_One_hundred_and_seven), ((5560,4368),He
ather_Three_hundred_and_eighty-four))
```

testroutesRDD.take(2)

val routes = testroutesRDD.map(route => ((idlist2(namelist2.indexOf(route.origin)),idlist2(namelist2.indexOf(route.dest))), route.route)).distinct

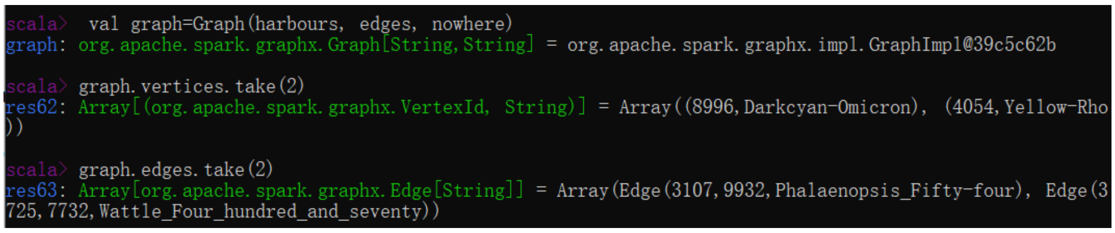
routes.take(2)



```
scala> val edges = routes.map{case((org_id, dest_id), route_name) =>Edge(org_id, dest_id, route_name)}
edges: org.apache.spark.rdd.RDD[org.apache.spark.graphx.Edge[String]] = MapPartitionsRDD[112] at map at <c
onsole>:30

scala> edges.take(1)
res60: Array[org.apache.spark.graphx.Edge[String]] = Array(Edge(9618,3394,Scabiosa_One_hundred_and_seven))
```

val edges = routes.map{case((org_id, dest_id), route_name) =>Edge(org_id, dest_id, route_name)}

edges.take(1)



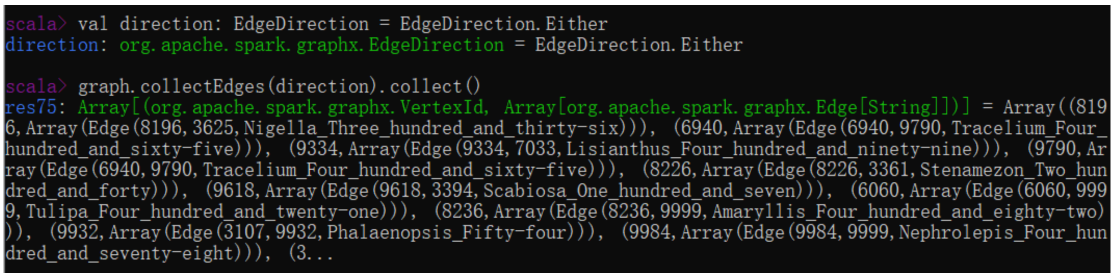
```
scala>  val graph=Graph(harbours, edges, nowhere)
graph: org.apache.spark.graphx.Graph[String,String] = org.apache.spark.graphx.impl.GraphImpl@39c5c62b

scala> graph.vertices.take(2)
res62: Array[(org.apache.spark.graphx.VertexId, String)] = Array((8996,Darkcyan-Omicron), (4054,Yellow-Rho
))

scala> graph.edges.take(2)
res63: Array[org.apache.spark.graphx.Edge[String]] = Array(Edge(3107,9932,Phalaenopsis_Fifty-four), Edge(3
725,7732,Wattle_Four_hundred_and_seventy))
```

val graph=Graph(harbours, edges, nowhere)

graph.vertices.take(2)

graph.edges.take(2)

Q2

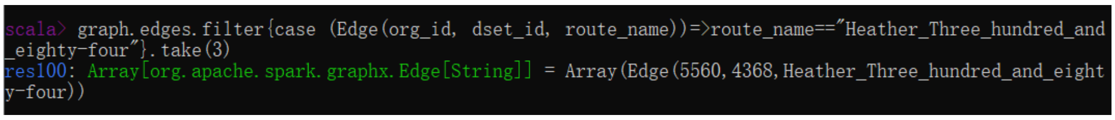


```
scala> val direction: EdgeDirection = EdgeDirection.Either
direction: org.apache.spark.graphx.EdgeDirection = EdgeDirection.Either

scala> graph.collectEdges(direction).collect()
res75: Array[(org.apache.spark.graphx.VertexId, Array[org.apache.spark.graphx.Edge[String]])] = Array((819
6,Array(Edge(8196,3625,Nigella_Three_hundred_and_thirty-six))), (6940,Array(Edge(6940,9790,Tracelium_Four_
hundred_and_sixty-five))), (9334,Array(Edge(9334,7033,Lisianthus_Four_hundred_and_ninety-nine))), (9790,Ar
ray(Edge(6940,9790,Tracelium_Four_hundred_and_sixty-five))), (8226,Array(Edge(8226,3361,Stenamezon_Two_hun
dred_and_forty))), (9618,Array(Edge(9618,3394,Scabiosa_One_hundred_and_seven))), (6060,Array(Edge(6060,999
9,Tulipa_Four_hundred_and_twenty-one))), (8236,Array(Edge(8236,9999,Amaryllis_Four_hundred_and_eighty-two)
)), (9932,Array(Edge(3107,9932,Phalaenopsis_Fifty-four))), (9984,Array(Edge(9984,9999,Nephrolepis_Four_hun
dred_and_seventy-eight))), (3...
```

val direction: EdgeDirection = EdgeDirection.Either

graph.collectEdges(direction).collect()

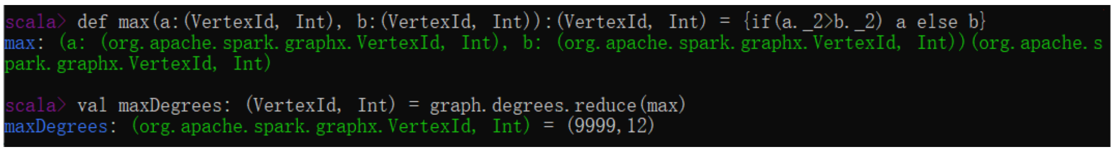
Q3



```
scala> graph.edges.filter{case (Edge(org_id, dset_id, route_name))=>route_name=="Heather_Three_hundred_and
_eighty-four"}.take(3)
res100: Array[org.apache.spark.graphx.Edge[String]] = Array(Edge(5560,4368,Heather_Three_hundred_and_eight
y-four))
```

graph.edges.filter{case (Edge(org_id, dset_id, route_name))=>route_name=="Heather_Three_hundred_and
_eighty-four"}.take(3)

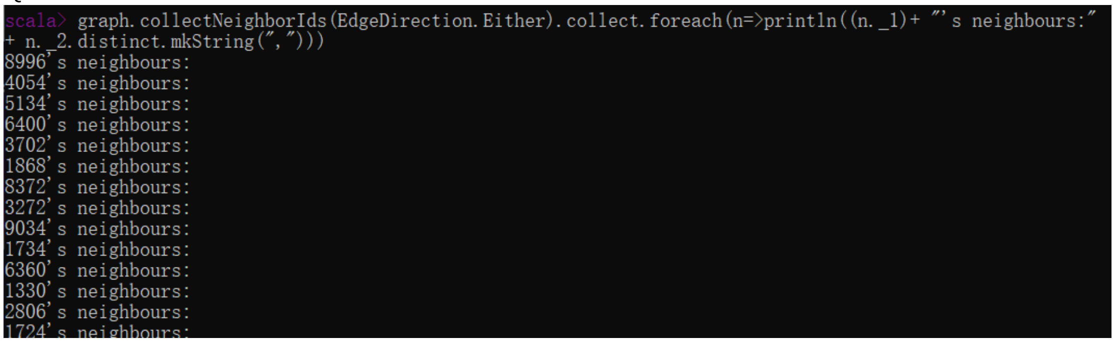
Q4



```
scala> def max(a:(VertexId, Int), b:(VertexId, Int)):(VertexId, Int) = {if(a._2>b._2) a else b}
max: (a: (org.apache.spark.graphx.VertexId, Int), b: (org.apache.spark.graphx.VertexId, Int))(org.apache.s
park.graphx.VertexId, Int)

scala> val maxDegrees: (VertexId, Int) = graph.degrees.reduce(max)
maxDegrees: (org.apache.spark.graphx.VertexId, Int) = (9999,12)
```

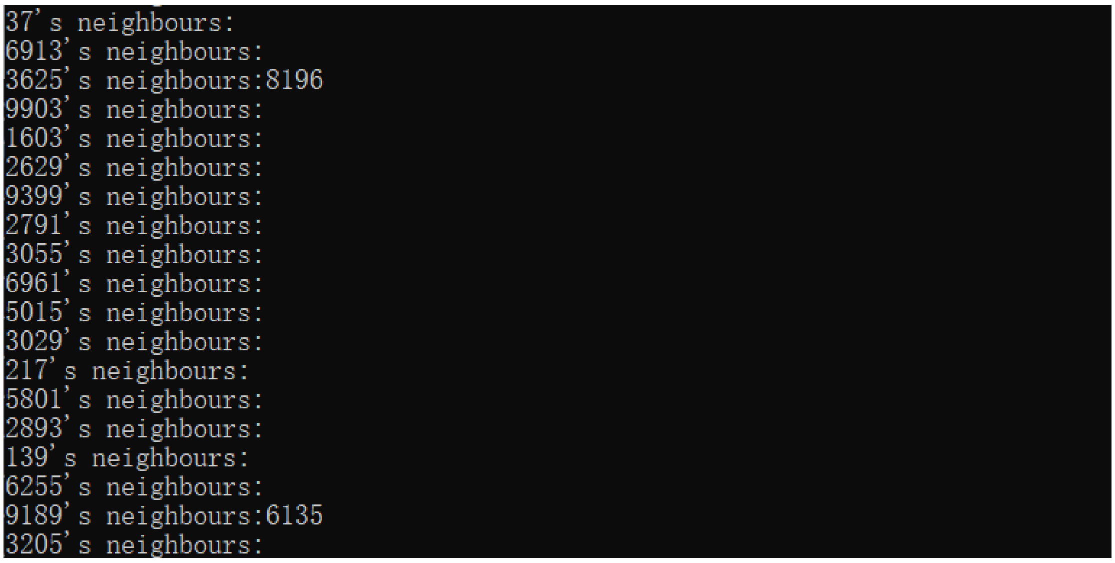
def max(a:(VertexId, Int), b:(VertexId, Int)):(VertexId, Int) = {if(a._2>b._2) a else b}
val maxDegrees: (VertexId, Int) = graph.degrees.reduce(max)

Q5

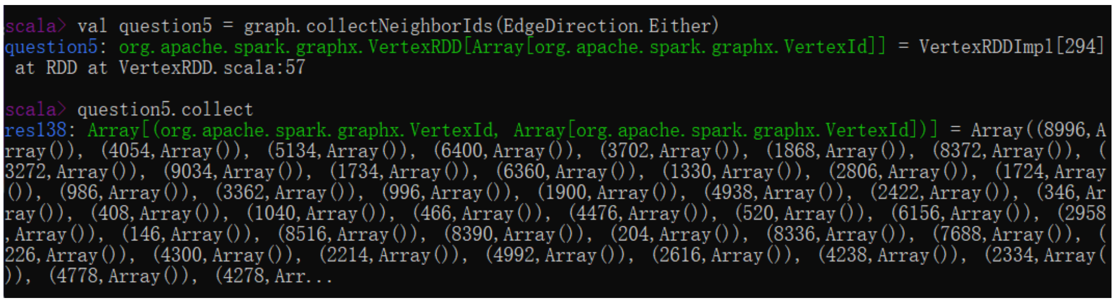


```
scala> graph.collectNeighborIds(EdgeDirection.Either).collect.foreach(n=>println((n._1)+ "'s neighbours:"
+ n._2.distinct.mkString(",")))
8996's neighbours:
4054's neighbours:
5134's neighbours:
6400's neighbours:
3702's neighbours:
1868's neighbours:
8372's neighbours:
3272's neighbours:
9034's neighbours:
1734's neighbours:
6360's neighbours:
1330's neighbours:
2806's neighbours:
1724's neighbours:
```

graph.collectNeighborIds(EdgeDirection.Either).collect.foreach(n=>println((n._1)+ "'s neighbours:"+ n._2.distinct.mkString(",")))

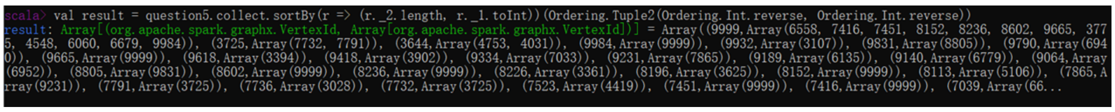


```
37's neighbours:
6913's neighbours:
3625's neighbours:8196
9903's neighbours:
1603's neighbours:
2629's neighbours:
9399's neighbours:
2791's neighbours:
3055's neighbours:
6961's neighbours:
5015's neighbours:
3029's neighbours:
217's neighbours:
5801's neighbours:
2893's neighbours:
139's neighbours:
6255's neighbours:
9189's neighbours:6135
3205's neighbours:
```



```
scala> val question5 = graph.collectNeighborIds(EdgeDirection.Either)
question5: org.apache.spark.graphx.VertexRDD[Array[org.apache.spark.graphx.VertexId]] = VertexRDDImpl[294]
 at RDD at VertexRDD.scala:57

scala> question5.collect
res138: Array[(org.apache.spark.graphx.VertexId, Array[org.apache.spark.graphx.VertexId])] = Array((8996,A
rray()), (4054,Array()), (5134,Array()), (6400,Array()), (3702,Array()), (1868,Array()), (8372,Array()), (
3272,Array()), (9034,Array()), (1734,Array()), (6360,Array()), (1330,Array()), (2806,Array()), (1724,Array
()), (986,Array()), (3362,Array()), (996,Array()), (1900,Array()), (4938,Array()), (2422,Array()), (346,Ar
ray()), (408,Array()), (1040,Array()), (466,Array()), (4476,Array()), (520,Array()), (6156,Array()), (2958
,Array()), (146,Array()), (8516,Array()), (8390,Array()), (204,Array()), (8336,Array()), (7688,Array()), (
226,Array()), (4300,Array()), (2214,Array()), (4992,Array()), (2616,Array()), (4238,Array()), (2334,Array(
)), (4778,Array()), (4278,Arr...
```

val question5 = graph.collectNeighborIds(EdgeDirection.Either)
question5.collect



```
scala> val result = question5.collect.sortBy(r => (r._2.length, r._1.toInt))(Ordering.Tuple2(Ordering.Int.reverse, Ordering.Int.reverse))
result: Array[(org.apache.spark.graphx.VertexId, Array[org.apache.spark.graphx.VertexId])] = Array((9999,Array(6558, 7416, 7451, 8152, 8236, 8602, 9665, 377
5, 4548, 6060, 6679, 9984)), (3725,Array(7732, 7791)), (3644,Array(4753, 4031)), (9984,Array(9999)), (9932,Array(3107)), (9831,Array(8805)), (9790,Array(694
0)), (9665,Array(9999)), (9618,Array(3394)), (9418,Array(3902)), (9334,Array(7033)), (9231,Array(7865)), (9189,Array(6135)), (9140,Array(6779)), (9064,Array
(6952)), (8805,Array(9831)), (8602,Array(9999)), (8236,Array(9999)), (8226,Array(3361)), (8196,Array(3625)), (8152,Array(9999)), (8113,Array(5106)), (7865,A
rray(9231)), (7791,Array(3725)), (7736,Array(3028)), (7732,Array(3725)), (7523,Array(4419)), (7451,Array(9999)), (7416,Array(9999)), (7039,Array(66...
```

val result = question5.collect.sortBy(r => (r._2.length, r._1.toInt))(Ordering.Tuple2(Ordering.Int.reverse, Ordering.Int.reverse))