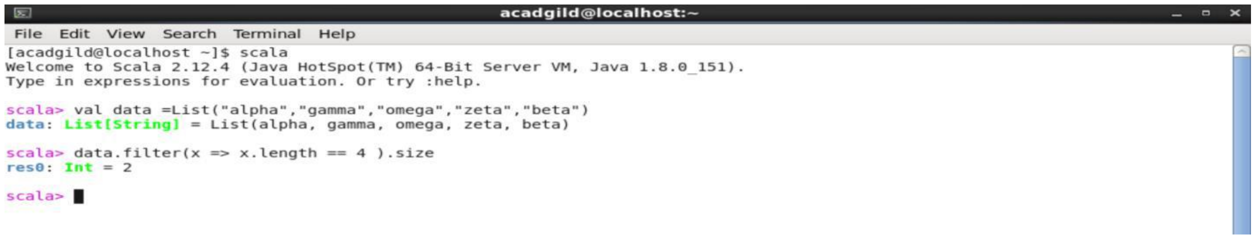
Session 14: Scala Basics 1

Assignment 1

Task 1

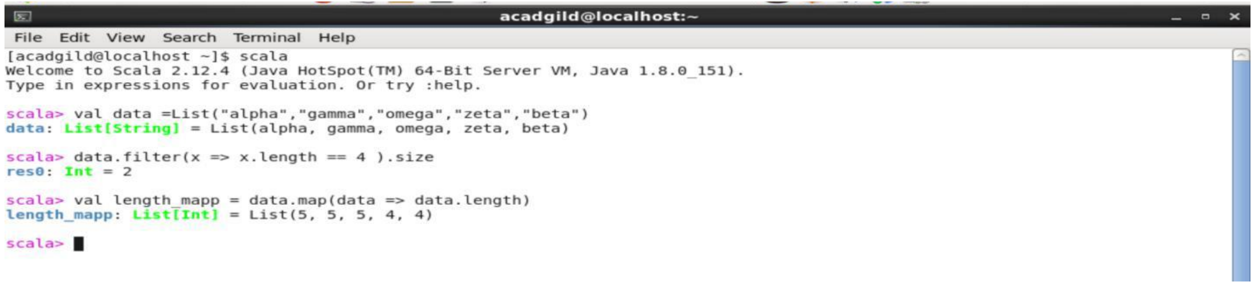
Given a list of strings – List[String] ("alpha", "gamma", "omega", ''zeta", "beta")

- Find the count of all strings with length 4



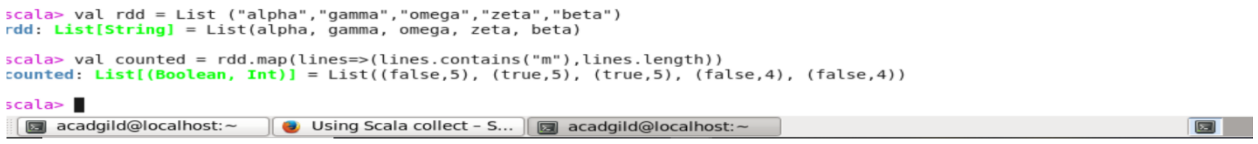
```
[acadgild@localhost ~]$ scala
Welcome to Scala 2.12.4 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_151).
Type in expressions for evaluation. Or try :help.

scala> val data =List("alpha","gamma","omega","zeta","beta")
data: List[String] = List(alpha, gamma, omega, zeta, beta)

scala> data.filter(x => x.length == 4 ).size
res0: Int = 2

scala> 
```

- Convert the list of strings to a list of integers, where each string is mapped to its corresponding length.



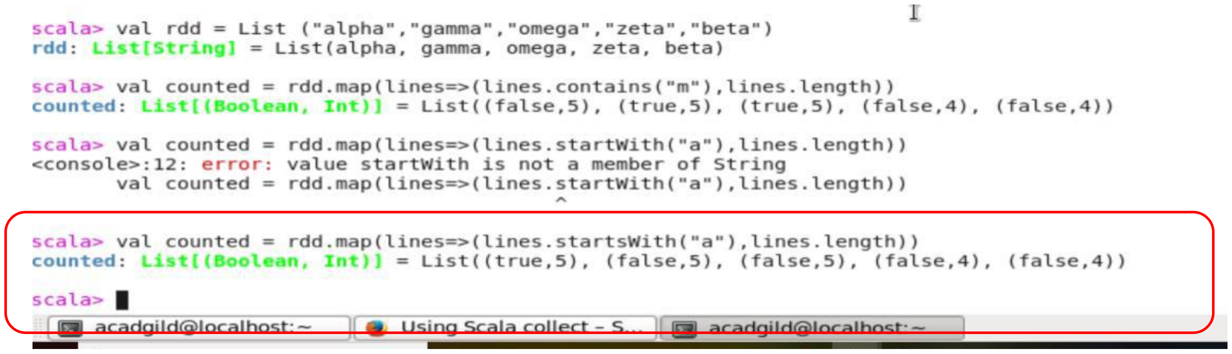
```
[acadgild@localhost ~]$ scala
Welcome to Scala 2.12.4 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_151).
Type in expressions for evaluation. Or try :help.

scala> val data =List("alpha","gamma","omega","zeta","beta")
data: List[String] = List(alpha, gamma, omega, zeta, beta)

scala> data.filter(x => x.length == 4 ).size
res0: Int = 2

scala> val length_mapp = data.map(data => data.length)
length_mapp: List[Int] = List(5, 5, 5, 4, 4)

scala> 
```

- Find count of all strings which contain alphabet 'm'.
  This shows two values contains letter 'm' i.e. 'gamma' and 'omega'.



```
scala> val rdd = List ("alpha","gamma","omega","zeta","beta")
rdd: List[String] = List(alpha, gamma, omega, zeta, beta)

scala> val counted = rdd.map(lines=>(lines.contains("m"),lines.length))
counted: List[(Boolean, Int)] = List((false,5), (true,5), (true,5), (false,4), (false,4))

scala> 
```

- Find the count of all strings which start with the alphabet ‘a’.
  This shows one value starts with ‘a’ i.e. ‘alpha’



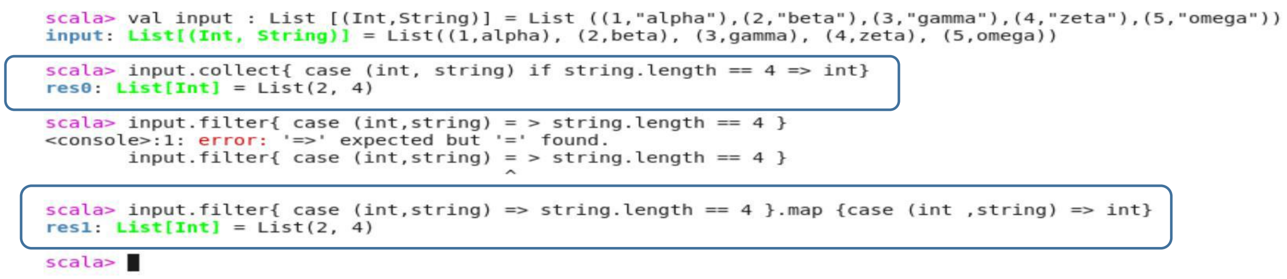
```
scala> val rdd = List ("alpha","gamma","omega","zeta","beta")
rdd: List[String] = List(alpha, gamma, omega, zeta, beta)

scala> val counted = rdd.map(lines=>(lines.contains("m"),lines.length))
counted: List[(Boolean, Int)] = List((false,5), (true,5), (true,5), (false,4), (false,4))

scala> val counted = rdd.map(lines=>(lines.startWith("a"),lines.length))
<console>:12: error: value startWith is not a member of String
       val counted = rdd.map(lines=>(lines.startWith("a"),lines.length))
                                           ^

scala> val counted = rdd.map(lines=>(lines.startsWith("a"),lines.length))
counted: List[(Boolean, Int)] = List((true,5), (false,5), (false,5), (false,4), (false,4))

scala> █
```

Task 2

Create a list of tuples, where the 1st element of the tuple is an int and the second element is a string.

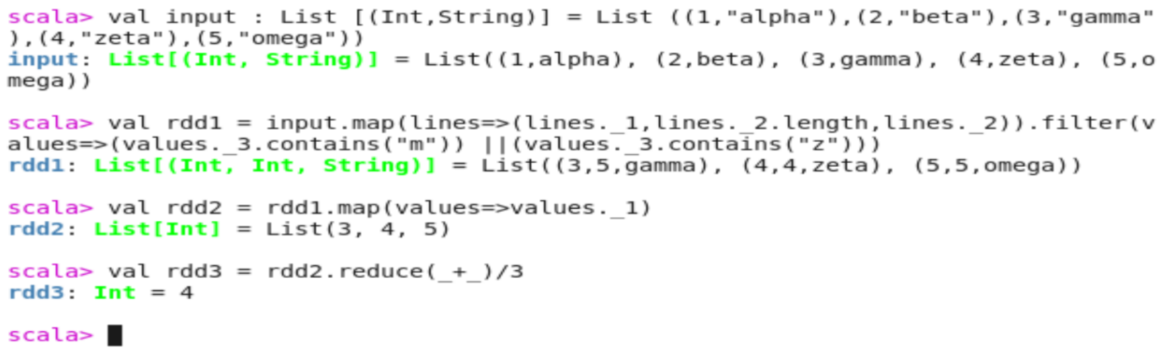
Example – ((1,’alpha’), (2,’beta’),(3,’gamma’),(4,’zeta’),(5,’omega’))

- For the above list print the numbers where the corresponding string length is 4.
  Here the 2nd and 4th elements length is equal to 4



```
scala> val input : List [(Int,String)] = List ((1,"alpha"),(2,"beta"),(3,"gamma"),(4,"zeta"),(5,"omega"))
input: List[(Int, String)] = List((1,alpha), (2,beta), (3,gamma), (4,zeta), (5,omega))

scala> input.collect{ case (int, string) if string.length == 4 => int}
res0: List[Int] = List(2, 4)

scala> input.filter{ case (int,string) = > string.length == 4 }
<console>:1: error: '=>' expected but '=' found.
       input.filter{ case (int,string) = > string.length == 4 }
                                       ^

scala> input.filter{ case (int,string) => string.length == 4 }.map {case (int ,string) => int}
res1: List[Int] = List(2, 4)

scala> █
```

- Find the average of all numbers, where the corresponding string contains alphabet ‘m’ or alphabet ‘z’.



```
scala> val input : List [(Int,String)] = List ((1,"alpha"),(2,"beta"),(3,"gamma"
),(4,"zeta"),(5,"omega"))
input: List[(Int, String)] = List((1,alpha), (2,beta), (3,gamma), (4,zeta), (5,o
mega))

scala> val rdd1 = input.map(lines=>(lines._1,lines._2.length,lines._2)).filter(v
alues=>(values._3.contains("m")) ||(values._3.contains("z")))
rdd1: List[(Int, Int, String)] = List((3,5,gamma), (4,4,zeta), (5,5,omega))

scala> val rdd2 = rdd1.map(values=>values._1)
rdd2: List[Int] = List(3, 4, 5)

scala> val rdd3 = rdd2.reduce(_+_)/3
rdd3: Int = 4

scala> 
```