```
package week6
import scala.io.Source
import scala.collection.immutable.List
object lecture7 {
  println("Welcome to the Scala worksheet1")      //> Welcome to the Scala worksheet1

  val in = Source.fromURL("http://lamp.epfl.ch/files/content/sites/lamp/files/teaching/progfun/linuxwords.txt")
                                                  //> in  : scala.io.BufferedSource = non-empty iterator

  val words = in.getLines.toList.filter(word=>word.forall(lt=>lt.isLetter))
                                                  //> words  : List[String] = List(Aarhus, Aaron, Ababa, aback, abaft, abandon, ab
                                                  //| andoned, abandoning, abandonment, abandons, abase, abased, abasement, abasem
                                                  //| ents, abases, abash, abashed, abashes, abashing, abasing, abate, abated, aba
                                                  //| tement, abatements, abater, abates, abating, Abba, abbe, abbey, abbeys, abbo
                                                  //| t, abbots, Abbott, abbreviate, abbreviated, abbreviates, abbreviating, abbre
                                                  //| viation, abbreviations, Abby, abdomen, abdomens, abdominal, abduct, abducted
                                                  //| , abduction, abductions, abductor, abductors, abducts, Abe, abed, Abel, Abel
                                                  //| ian, Abelson, Aberdeen, Abernathy, aberrant, aberration, aberrations, abet,
                                                  //| abets, abetted, abetter, abetting, abeyance, abhor, abhorred, abhorrent, abh
                                                  //| orrer, abhorring, abhors, abide, abided, abides, abiding, Abidjan, Abigail,
                                                  //| Abilene, abilities, ability, abject, abjection, abjections, abjectly, abject
                                                  //| ness, abjure, abjured, abjures, abjuring, ablate, ablated, ablates, ablating
                                                  //| , ablation, ablative, ab
                                                  //| Output exceeds cutoff limit.

  val mnem = Map('2'->"ABC",'3'->"DEF",'4'->"GHI",'5'->"JKL",
                  '6'->"MNO",'7'->"PQRS",'8'->"TUV",'9'->"WXYZ")
                                                  //> mnem  : 
scala.collection.immutable.Map[Char,String] = Map(8 -> TUV, 4 -> GHI
                                                  //| , 9 -> WXYZ, 5 
-> JKL, 6 -> MNO, 2 -> ABC, 7 -> PQRS, 3 -> DEF)

  /**Invert the mnem map to give a map from chars 'A'...'Z' to 
'2'...'9'*/

  val charCode:Map[Char,Char] =for((digit,letters)<-mnem; letter<-
letters) yield letter -> digit
                                                  //> charCode  : 
Map[Char,Char] = Map(E -> 3, X -> 9, N -> 6, T -> 8, Y -> 9, J -
                                                  //| > 5, U -> 8, F 
-> 3, A -> 2, M -> 6, I -> 4, G -> 4, V -> 8, Q -> 7, L -> 5, 
                                                  //|  B -> 2, P -> 
7, C -> 2, H -> 4, W -> 9, K -> 5, R -> 7, O -> 6, D -> 3, Z -
                                                  //| > 9, S -> 7)

  /**Maps a word to the digit string it can represent, e.g. "Java"-
>"5282"*/
  def wordCode(word:String):String = 
word.toUpperCase.map(x=>charCode(x)) //可以简写为 .map(charCode),因为
map就是一函数。
                                                  //> wordCode: 
(word: String)String

  wordCode("Java")                                //> res0: String = 
5282



  /**
  *A map from digit strings to the words that represent them
  *e.g. "5282" -> List("Java","Kata","Lava",...)
  *Note: A missing nuber should map tot he empty set,e.g. "1111" -> 
List()
  */

 val wordsForNum:Map[String,Seq[String]] = words.groupBy(wordCode) 
withDefaultValue(Seq())
                                                  //> wordsForNum  : 
```

```
Map[String,Seq[String]] = Map(63972278 -> List(newscast), 29
                                                  //| 237638427 ->
List(cybernetics), 782754448 -> List(starlight), 2559464 -> Li
                                                  //| st(allying),
862532733 -> List(uncleared), 365692259 -> List(enjoyably), 86
                                                  //| 8437 ->
List(unties), 33767833 -> List(deportee), 742533 -> List(picked), 3
                                                  //| 364646489 ->
List(femininity), 3987267346279 -> List(extraordinary), 785539
                                                  //| 7 ->
List(pulleys), 67846493 -> List(optimize), 4723837 -> List(grafter),
3
                                                  //| 86583 ->
List(evolve), 78475464 -> List(Stirling), 746459 -> List(singly),
                                                  //| 847827 ->
List(vistas), 546637737 -> List(lionesses), 28754283 -> List(curl
                                                  //| icue),
84863372658 -> List(thunderbolt), 46767833 -> List(imported), 264374
                                                  //| 64 ->
List(angering, cohering), 8872267 -> List(turbans), 77665377 ->
List(
                                                  //| spoolers),
46636233 -> List(homemade), 7446768759 -> List(rigorously), 7464
                                                  //| 4647 ->
List(ringings), 633738 -> List(offset), 847825 -> List(visual), 772
                                                  //| 832 ->
List(Pravda), 47
                                                  //| Output exceeds
cutoff limit.
/**Return all ways to encode a number as a list of words*/
def encode(number:String):Set[List[String]] =
if(number.isEmpty) Set(List())
else{
 for{
    split <- 1 to number.length
    word <- wordsForNum(number.take(split))
    rest <- encode(number.drop(split))
  } yield word::rest
  }.toSet                                         //> encode:
(number: String)Set[List[String]]

  encode("7225247386")                            //> res1:
Set[List[String]] = Set(List(rack, ah, re, to), List(sack, ah, re,
to
```

```
                                                  //| ), List(Scala, ire, to), List(sack, air, fun), List(rack, air, fun), List(r
                                                  //| ack, bird, to), List(pack, air, fun), List(pack, ah, re, to), List(pack, bi
                                                  //| rd, to), List(Scala, is, fun), List(sack, bird, to))
  def translate(number:String):Set[String] =encode(number).map(_.mkString(" "))
                                                  //> translate: (number: String)Set[String]
  translate("7225247386")                         //| , sack ah re to, rack air fun)
}
```