import scala.annotation.tailrec  
  
object Assignment1 {  
 val days = List("Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday")  
 val products = Map("fruits" -> 1000, "vegetables" -> 500)  
  
 def main(args: Array[String]): Unit = {  
 println("Task\_1:")  
 println(Task\_1("a"))  
 println(Task\_1("b"))  
 println(Task\_1("c"))  
 println("Task\_2:")  
 println(Task\_2("a", days))  
 println(Task\_2("b", days))  
 println("Task\_3:")  
 println(Task\_3(days))  
 println("Task\_4:")  
 println(Task\_4("a"))  
 println(Task\_4("b"))  
 println(Task\_4("c"))  
 println("Task\_5:")  
 println(Task\_5)  
 println("Task\_6:")  
 println(Task\_6(1 :: 2 :: 3 :: Nil))  
 println("Task\_7")  
 println(Task\_7(-11 :: -2.4 :: 3.0 :: 20.3 :: Nil))  
 println("Task\_8")  
 Task\_8(Tuple3(1, 300.2, "abc"))  
 println("Task\_9")  
 println(Task\_9(-1.3 :: 2.1 :: 0.0 :: 3.4 :: 0.0 :: Nil))  
 println("Task\_10")  
 Task\_10("vegetables")  
 Task\_10("Fruits")  
 Task\_10("apple")  
  
 }  
  
 def Task\_1(t: String): String = {  
 var s: String = ""  
 if (t.equals("a"))  
 for (day <- days)  
 s += day + ", "  
 else if (t.equals("b"))  
 for (day <- days if day.toLowerCase.startsWith("s"))  
 s += day + ", "  
 else if (t.equals("c")) {  
 var i: Int = 0  
 while (i < 7) {  
 s += days(i) + ", "  
 i += 1  
 }  
 }  
 return s.substring(0, s.length - 2)  
 }  
  
 def Task\_2(t: String, l: List[String]): String = {  
 if (t == "a") {  
 if (l.tail.isEmpty)  
 return l.head  
 return l.head + ", " + Task\_2("a", l.tail)  
 }  
 if (t == "b") {  
 if (l.tail.isEmpty)  
 return l.head  
 return Task\_2("a", l.tail) + ", " + l.head  
 } else return ""  
 }  
 def Task\_3(ll: List[String]): String = {  
 @tailrec  
 def Task\_3(l: List[String],s:String): String = {  
 if (l.tail.isEmpty)  
 return s+l.head  
 else Task\_3(l.tail,s+l.head + ", ")  
 }  
 Task\_3(ll,"")  
 }  
  
 def Task\_4(t: String): String = {  
 var s: String = ""  
 if (t.equals("a"))  
 s = days.foldLeft("")(\_ + \_ + ", ")  
 else if (t.equals("b"))  
 s = days.foldRight("")(\_ + ", " + \_)  
 else if (t.equals("c"))  
 s = days.foldRight("") { (next, sum) => if (next.toLowerCase.startsWith("s")) next + ", " + sum else sum }  
 return s.substring(0, s.length - 2)  
 }  
  
 def Task\_5: Map[String, Double] = products.mapValues(\_ \* 0.9)  
  
 def Task\_6(l: List[Int]): List[Int] = l.map(\_ + 1)  
  
 def Task\_7(l: List[Double]): List[Double] = l.filter(-5 < \_).filter(\_ < 12).map(\_.abs)  
  
 def Task\_8(tuple: Tuple3[Int, Double, String]): Unit = println(tuple)  
  
 def Task\_9(l: List[Double]): List[Double] = {  
 if (l.isEmpty) l  
 else if (l.head == 0) Task\_9(l.tail)  
 else l.head :: Task\_9(l.tail)  
 }  
  
 def Task\_10(product: String) = {  
 val price: Option[Int] = products.get(product.toLowerCase)  
 println(price.getOrElse(" We don't have such items -> " + product))  
 if (price.isDefined && price.get >= 1000)  
 println(product + " put it in your bag and pay for it")  
 println("#################################")  
 }  
  
}