import java.util.regex.Pattern  
  
// Second Solution  
  
class Day11 : Day<String?> {  
 fun containsStraight(candidate: String?): Boolean {  
 return Pattern.matches(".\*(abc|bcd|cde|def|efg|fgh|pqr|qrs|rst|stu|tuv|uvw|vwx|wxy|xyz).\*", candidate)  
 }  
  
 fun noForbiddenChars(candidate: String?): Boolean {  
 return !Pattern.matches(".\*[iol].\*", candidate)  
 }  
  
 fun containsTwoPairs(candidate: String?): Boolean {  
 return Pattern.matches(".\*(.)\\1.\*(.)\\2.\*", candidate)  
 }  
  
 fun isValid(candidate: String?): Boolean {  
 return containsStraight(candidate) &&  
 noForbiddenChars(candidate) &&  
 containsTwoPairs(candidate)  
 }  
  
 fun increment(candidate: String?): String {  
 return java.lang.Long.toString(candidate!!.*toLong*(36) + 1, 36).*replace*('0', 'a')  
 }  
  
 override fun part1(input: List<String?>?): String {  
 var password = increment(input!![0])  
 while (!isValid(password)) {  
 password = increment(password)  
 }  
 return password  
 }  
  
 override fun part2(input: List<String?>?): String {  
 var password = increment(input!![0])  
 while (!isValid(password)) {  
 password = increment(password)  
 }  
 password = increment(password)  
 while (!isValid(password)) {  
 password = increment(password)  
 }  
 return password  
 }  
}

interface Day<T> {  
 fun part1(input: List<String?>?): T  
 fun part2(input: List<String?>?): T  
}