*/\*\*  
 \* Created by Shila Parajuli on 6/9/2016.  
 \*/***public class** Shakespeare {  
 **private static** def *wordList* = []  
 **private static** def *wordMap* = [:]  
  
 **public static void** main(String[] args) {  
 String fileContents;  
 **try** {  
 *//to read the dictionary file of shakespeare.txth* fileContents = **new** File(**'/home/mohan/shakespeare.txt'**).text  
 }  
 **catch** (FileNotFoundException e) {  
 println(**"File not found, pelase check the location"**)  
 }  
 *// separate the content of the file according to the given parameters  
 wordList* = fileContents.tokenize(**' \n\n,.:;!?'**)*//separate words from text and put it in a list  
 wordMap* = [:] *//map of words with their frequency  
 wordList*.each {  
 **if** (*wordMap*.containsKey(it.toLowerCase())) {  
 *wordMap*[it.toLowerCase()] += 1  
 } **else** {  
 *wordMap*[it.toLowerCase()] = 1  
 }  
 }  
 *wordMap* = *wordMap*.sort { -it.value }  
 def frequent = []  
 def counter = 1  
 *wordMap*.each {  
 **if** (counter == 21) {  
 **return true** }  
 def temp = []  
 temp.add(counter)  
 temp.add(it.getKey())  
 temp.add(it.getValue())  
 frequent.add(temp)  
 counter++  
 **return false** }  
 println(**"Top 20 frequent words with rank, word and frequency respectively: "**)  
 println(frequent)  
  
  
  
 println(**"Probability of words:"**)  
  
 println(**"Probability of \"THE\" "** + *takeData*([**"the"**]))  
 println(**"Probability of \"BECOME\" "** + *takeData*([**"become"**]))  
 println(**"Probability of \"BRAVE\" "** + *takeData*([**"brave"**]))  
 println(**"Probability of \"TREASON\" "** + *takeData*([**"treason"**]))  
  
 println()  
 println(**"Part B "**)  
 println(**"P[court|the] "** + *takeData*([**"the"**, **"court"**]))  
 println(**"P[word|his] "** + *takeData*([**"his"**, **"word"**]))  
 println(**"P[qualities|rare] "** + *takeData*([**"rare"**, **"qualities"**]))  
 println(**"P[men|young] "** + *takeData*([**"young"**, **"men"**]))  
  
 println()  
 println(**"Chain rule- dependent case:"**)  
 println(**"Probability of P[have, sent] "** + *takeData*([**"have"**, **"sent"**]))  
 println(**"Probability of P[will, look, upon] "** + *takeData*([**"will"**, **"look"**, **"upon"**]))  
 println(**"Probability of P[i, am, no, baby] "** + *takeData*([**"i"**, **"am"**, **"no"**, **"baby"**]))  
 println(**"Probability of P[wherefore, art, thou, romeo] "** + *takeData*([**"wherefore"**, **"art"**, **"thou"**, **"romeo"**]))  
  
 println()  
 println(**"Chain rule- independent case:"**)  
 println(**"Probability of P[have, sent] "** + *independentProbability*([**"have"**, **"sent"**]))  
 println(**"Probability of P[will, look, upon] "** + *independentProbability*([**"will"**, **"look"**, **"upon"**]))  
 println(**"Probability of P[i, am, no, baby] "** + *independentProbability*([**"i"**, **"am"**, **"no"**, **"baby"**]))  
 println(**"Probability of P[wherefore, art, thou, romeo] "** + *independentProbability*([**"wherefore"**, **"art"**, **"thou"**, **"romeo"**]))  
 }  
  
 *//this function calculates the probability for the most frequent word-pairs (bigrams) model* **public static float** takeData(wordForProb) {  
 def probab = 0  
 **switch** (wordForProb.size()) {  
 **case** 1:  
 probab = *wordMap*.get(wordForProb[0]) / *wordList*.size()  
 **break  
 case** 2: *// if it is bi-gram model* probab = *calculateProbability*(wordForProb)  
 **break  
 case** 3: *// if it is tri-gram model* probab = *wordMap*.get(wordForProb[0]) / *wordList*.size() \* *calculateProbability*([wordForProb[0], wordForProb[1]]) \*  
 *calculateProbability*([wordForProb[1], wordForProb[2]])  
 **break  
 case** 4: *// if it is 4-gram model* probab = *wordMap*.get(wordForProb[0]) / *wordList*.size() \* *calculateProbability*([wordForProb[0], wordForProb[1]]) \*  
 *calculateProbability*([wordForProb[1], wordForProb[2]],) \*  
 *calculateProbability*([wordForProb[2], wordForProb[3]])  
 **break** }  
 **return** probab  
 }  
  
 **public static float** calculateProbability(wordForProb) {*// takes two words in a list and find the probability* def counter = 0  
 def probab  
 **for** (**int** i = 0; i < *wordList*.size - 2; i++) {  
 def firstWord = *wordList*[i]  
 def secondWord = *wordList*[i + 1]  
 **if** (firstWord == wordForProb[0] && secondWord == wordForProb[1]) {  
 counter++  
 }  
 }  
 probab = counter / (*wordMap*.get(wordForProb[0]))  
 **return** probab  
 }  
  
  
 **public static float** independentProbability(*wordList*) {  
 def ans = 1  
 *wordList*.each {  
 ans \*= *takeData*([it])  
 }  
 **return** ans  
 }  
 }