German University in Cairo
Media Engineering and Technology
Prof. Dr. Slim Abdennadher
Dr. Nada Sharaf

Concepts of Programming Languages, Spring term 2020
TestCases
"Automatic Text Generation"

wordToken

```
>  wordToken "the sun is shining. the wind is blowing."
["the","sun","is","shining",".","the","wind","is","blowing","."]
```

```
>  wordToken "hello! how are you?"
["hello","!","how","are","you","?"]
```

wordTokenList

```
> wordTokenList ["the man is the man. he is great","the man saw the saw."]
["the","man","is","the","man",".","he","is","great","the","man","saw","the","saw","."]

> wordTokenList ["hello! how are you?","the sun is shining. the wind is
↪  blowing.","amazing","sorry about that"]

["hello","!","how","are","you","?","the","sun","is","shining",".","the","wind","is"," ⌋
↪  blowing",".","amazing","sorry","about","that"]
```

uniqueBigrams

```
> uniqueBigrams ["the","man","is","the","man","."]
[("man","is"),("is","the"),("the","man"),("man",".")]

> uniqueBigrams ["you","have","not","left","enough","space","between","pig","and","an ⌋
↪  d","and","and","and","whistle","."]

[("you","have"),("have","not"),("not","left"),("left","enough"),("enough","space"),(" ⌋
↪  space","between"),("between","pig"),("pig","and"),("and","and"),("and","whistle") ⌋
↪  ,("whistle",".")]
```

uniqueTrigrams

```
> uniqueTrigrams ["the","man","is","the","man","."]
[("the","man","is"),("man","is","the"),("is","the","man"),("the","man",".")]
```

```
> uniqueTrigrams ["you","have","not","left","enough","space","between","pig","and","a⌋
↪  nd","and","and","and","whistle","."]

[("you","have","not"),("have","not","left"),("not","left","enough"),("left","enough", ⌋
↪  "space"),("enough","space","between"),("space","between","pig"),("between","pig", ⌋
↪  "and"),("pig","and","and"),("and","and","and"),("and","and","whistle"),("and","wh⌋
↪  istle","."))]
```

bigramsFreq

```
> bigramsFreq ["the","man","is","the","man","."]
[(("man","is"),1),(("is","the"),1),(("the","man"),2),(("man","."),1)]

> bigramsFreq ["you","have","not","left","enough","space","between","pig","and","and"⌋
↪  ,"and","and","and","whistle","."]

[(("you","have"),1),(("have","not"),1),(("not","left"),1),(("left","enough"),1),(("en⌋
↪  ough","space"),1),(("space","between"),1),(("between","pig"),1),(("pig","and"),1)⌋
↪  ,(("and","and"),4),(("and","whistle"),1),(("whistle","."),1)]
```

trigramsFreq

```
> trigramsFreq ["the","man","is","the","man","."]
[(("the","man","is"),1),(("man","is","the"),1),(("is","the","man"),1),(("the","man","⌋
↪  ."),1)]

> trigramsFreq ["you","have","not","left","enough","space","between","pig","and","and⌋
↪  ","and","and","and","whistle","."]

[(("you","have","not"),1),(("have","not","left"),1),(("not","left","enough"),1),(("le⌋
↪  ft","enough","space"),1),(("enough","space","between"),1),(("space","between","pi⌋
↪  g"),1),(("between","pig","and"),1),(("pig","and","and"),1),(("and","and","and"),3⌋
↪  ),(("and","and","whistle"),1),(("and","whistle","."),1)]
```

getFreq

```
> getFreq 'a' [('f',1),('a',2),('b',1)]
2

> getFreq ("and","and","and") [(("pig","and","and"),1),(("and","and","and"),3),(("and⌋
↪  ","and","whistle"),1),(("and","whistle","."),1)]
3
```

generateOneProb

```
> generateOneProb (("the","man","is"),1)
[(("he","is"),1),(("is","great"),1),(("great","the"),1),(("the","man"),3)]
0.3333333333333333

> generateOneProb (("he","is","a"),2)
[(("he","is"),5),(("is","great"),1),(("great","the"),1),(("the","man"),3)]
0.4
```

genProbPairs

```
> genProbPairs
 ↪  [(("the","man","is"),1),(("man","is","the"),1),(("is","the","man"),1),(("the","ma⌋
 ↪  n","."),1),(("man",".","the"),1),((".","the","man"),1),(("the","man","saw"),1)]
 ↪  [(("man","is"),1),(("is","the"),1),(("man","."),1),((".","the"),1),(("the","man")⌋
 ↪  ,3),(("man","saw"),1)]

[(("the","man","is"),0.333333333333333),(("man","is","the"),1.0),(("is","the","man"),⌋
 ↪  1.0),(("the","man","."),0.333333333333333),(("man",".","the"),1.0),((".","the","m⌋
 ↪  an"),1.0),(("the","man","saw"),0.333333333333333)]
```

generateNextWord

The output is in random order and might be repeated

```
> generateNextWord ["the","man"][(("the","man","is"),0.006),(("man","is","the"),1.0),⌋
 ↪  (("is","the","man"),1.0),(("the","man","."),0.333333333333333),(("man",".","the")⌋
 ↪  ,1.0),((".","the","man"),1.0),(("the","man","saw"),0.333333333333333)]

"saw"

> generateNextWord ["the","man"][(("the","man","is"),0.006),(("man","is","the"),1.0),⌋
 ↪  (("is","the","man"),1.0),(("the","man","."),0.333333333333333),(("man",".","the")⌋
 ↪  ,1.0),((".","the","man"),1.0),(("the","man","saw"),0.333333333333333)]

"."
```

generateText

A possible output to the following (using the shorter list docs)

```
> generateText "the man" 2
"the man saw the"

> generateText "the man" 2
"the man. he"

> generateText "saw the" 2
Program error: Sorry, it is not possible to infer from current database
```

A possible output to the following (using the longer list docs)

```
> generateText "it is" 3
"it is at 49 degree"

> generateText "it is" 4
"it is at 51 degree 07"
```