

Question One

```
from collections import defaultdict
from turtle import Turtle, Screen

def word_freq_dict(file_name):
    """ Count the amount of times words appear in a string """

    file = open(file_name)
    readFile = file.read() # read file
    words = readFile.split() # splits string into words, puts each word as an
    element in a list
    word_dict = defaultdict(int) # dictionary for words to be placed in with
    amount of times they appear

    for word in words:
        word_dict[word] += 1 # adds item in 'words' to a dictionary and amount
    of times it appears

    return word_dict

def word_tower(turtle, screen, file_name):

    word_dict = word_freq_dict(file_name) # calls dictionary function

    turtle.up() # lift pen off page
    turtle.goto(0, - screen.window_height() // 2) # start at the center bottom
    of the page
    turtle.left(90) # point turtle up the page for subsequent forward() calls

    for key in sorted(word_dict, key=lambda k: word_dict[k], reverse=True): #
    sort words by frequency
        value = word_dict[key] * 15 # probably should compute this value based
    on # words and page height
        turtle.write(key, font=('Arial', value, 'normal'), align='center')
        turtle.forward(value)

yertle = Turtle()
yertle.hideturtle() # hide turtle image

screen = Screen()

word_tower(yertle, screen, 'data_file.txt')

screen.exitonclick()
```

Question Two

```
import random
import os
import glob
import sys
import errno
import nltk.data

path = '/home/clement/Documents/Frankenstein.txt'
files = glob.glob(path)

sentence_detector = nltk.data.load('tokenizers/punkt/dutch.pickle')
listOfSentences = []

for text in files:
    try:
        with open(text, 'rt', encoding='utf-8') as f:
            fullText = f.read()
            listOfSentences += [x.replace("\n", " ").replace(" ", " ").strip() for x
in random.sample(sentence_detector.tokenize(fullText), 100)]

    except IOError as exc:
        # Do not fail if a directory is found, just ignore it.
        if exc.errno != errno.EISDIR:
            raise

random_sample_input = random.sample(listOfSentences, 1)
print(random_sample_input)

# This block of code writes the result of the previous to a new file
random_sample_output = open("randomsample", "w", encoding='utf-8')
random_sample_input = map(lambda x: x+"\n", random_sample_input)
random_sample_output.writelines(random_sample_input)
random_sample_output.close()
```

Question Three

```
import sys

filename = sys.argv[1]
fp = open(filename)
data = fp.read()
words = data.split()
fp.close()

unwanted_chars = "., -_ (and so on)"
wordfreq = {}
for raw_word in words:
    word = raw_word.strip(unwanted_chars)
    if word not in wordfreq:
        wordfreq[word] = 0
    wordfreq[word] += 1

from collections import defaultdict
from turtle import Turtle, Screen

def word_freq_dict(file_name):
    """ Count the amount of times words appear in a string """

    file = open(file_name)
    readFile = file.read() # read file
```

```

    words = readFile.split() # splits string into words, puts each word as an
element in a list
    word_dict = defaultdict(int) # dictionary for words to be placed in with
amount of times they appear

    for word in words:
        word_dict[word] += 1 # adds item in 'words' to a dictionary and amount
of times it appears

    return word_dict

def word_tower(turtle, screen, file_name):

    word_dict = word_freq_dict(file_name) # calls dictionary function

    turtle.up() # lift pen off page
    turtle.goto(0, - screen.window_height() // 2) # start at the center bottom
of the page
    turtle.left(90) # point turtle up the page for subsequent forward() calls

    for key in sorted(word_dict, key=lambda k: word_dict[k], reverse=True): #
sort words by frequency
        value = word_dict[key] * 15 # probably should compute this value based
on # words and page height
        turtle.write(key, font=('Arial', value, 'normal'), align='center')
        turtle.forward(value)

yertle = Turtle()
yertle.hideturtle() # hide turtle image

screen = Screen()

word_tower(yertle, screen, 'data_file.txt')

screen.exitonclick()

import random
import os
import glob
import sys
import errno
import nltk.data

path = '/home/clement/Documents/Frankenstein.txt'
files = glob.glob(path)

sentence_detector = nltk.data.load('tokenizers/punkt/dutch.pickle')
listOfSentences = []

for text in files:
    try:
        with open(text, 'rt', encoding='utf-8') as f:
            fullText = f.read()
            listOfSentences += [x.replace("\n", " ").replace(" ", " ").strip() for x
in random.sample(sentence_detector.tokenize(fullText), 100)]

    except IOError as exc:
        # Do not fail if a directory is found, just ignore it.
        if exc.errno != errno.EISDIR:
            raise

random_sample_input = random.sample(listOfSentences, 1)
print(random_sample_input)

```

```
# This block of code writes the result of the previous to a new file
random_sample_output = open("randomsample", "w", encoding='utf-8')
random_sample_input = map(lambda x: x+"\n", random_sample_input)
random_sample_output.writelines(random_sample_input)
random_sample_output.close()
```