### Name: Tradd Schmidt

### This reflection is to be completed individually, though consultations with TAs and classmates are encouraged as long as they are appropriately acknowledged.

### The information stored in the Markov chains allows a program to generate text that "sound like" it was written by the one who wrote the text that was used to create the chains.

1. When we run *test\_Markov.py*, we ask the user for a file name and then call the **makeWordModel()** function. This function uses the Markov class to create a Markov chain that is referenced by "*tmpmodel*", a local variable. This chain is returned to the main program, and "*m*" now references that same Markov object.

Note that "model" and "state" are instance variables of the Markov class object.

* 1. Using the specific language of Python data structures, identify specifically what type of data structure or compound data structure "model" and "state" refer to. *Hint: Be specific! If one is a list containing another list which contains strings, say so!*

|  |
| --- |
| model: A dictionary where the keys are tuples containing strings and the items are lists containing one string  state: A tuple containing two strings |

* 1. Using only a single sentence for each, describe what each of these instance variables refers to after the call to the call **makeWordModel()**. *Hint: You can trace the code, use the watch in the debugger, or in print statements to see the contents of these.*

|  |
| --- |
| model: Refers to the entire dictionary of possible continuations of a sentence based on the previous word used and the current word being used  state: (None, None) |

1. Explain in your own words how the **add()** method modifies the "model" instance variable as the program processes the first 5 words in your *yourusername-csc236A6-input.txt* file.

|  |
| --- |
| It changes the state so that is on the next tuple combination |

1. Give an example from your Markov chain "model" instance variable where the list associated with a specific "state" contains more than one word. Explain the purpose of this list.

|  |
| --- |
| ('You', 'are'): ['ready', 'ready'] This has two readies because there are two instances where a ready follows are. |

1. Using example word string from your input file, explain in your own words what each line in the function **generateWordChain()** function does in the *test\_Markov.py* module.

|  |
| --- |
| words = [] # Creates an empty list for i in range(n): # a for loop to iterate the amount of times that the function call is given  next = markov.randomNext() # generates a random word from words that have not been used  if next is None: break # got to a final state   words.append(next) # adds the random word to the list return " ".join(words) # returns the list of words as a string |

1. Though this program is really fun to run, it clearly could be improved if the generated text did not frequently cut-off mid-sentence. Describe a design improvement which would serve to improve this deficiency.

|  |
| --- |
| When the list of words has been generated, the last element could be checked for punctuation. If there is no punctuation, you could iterate backwards until punctuation is found. Once it is found, the elements of the list after the element that has punctuation could be deleted and the list then be printed. |