Introduction

A Markov chain is a statistical model of a process. For certain kinds of processes, the next state of the process can be predicted solely from the current state, independently of previous history of the process. As an example, the board game "Chutes & Ladders" (or any game in which the moves are made solely by dice) forms a Markov chain. On the other hand, a card game such as blackjack is not a Markov chain because the next hand dealt depends on which cards have already been played.

Markov chains are used to represent arrivals and departures at airports and train stations, cruise control systems in vehicles, currency exchange rates, and more. The PageRank algorithm, at the heart of Google's search engine, is based on a Markov process.

One application of Markov chains is automated text generators[1]. Such an application takes a sample input text and builds a table that maps sequences of words to a list of words that follow the sequence in the sample input. The length of the sequence can vary. For example, consider the following:

```
I really like to eat burgers. I really like them with bacon. I eat burgers
with cheese, too. I eat them with lettuce, too.
```

Table construction uses a rolling window defined by the sequence length. If the sequence length is two, the sequence is called a "bi-gram." The table is initialized with a sequence of two "non-words" set aside to mark the beginning and end of the text. This example uses a newline character as a non-word. A table entry maps each bi-gram to a list of words that follow that bi-gram in the sample.

The initial prefix is two non-words. The first word in the sample is considered to "follow" this prefix.This is illustrated below:

\n	**\n**	I	really	like	to	eat	burgers.	I	really	like	them

The window is moved forward by one word and the new prefix is a non-word followed by the first word. The second word in the sample follows this prefix.

\n	**\n**	**I**	really	like	to	eat	burgers.	I	really	like	them

The process continues moving forward through the sample, recording what word follows each 2-word sequence in the sample.

\n	\n	**I**	**really**	like	to	eat	burgers.	I	really	like	them
\n	\n	I	**really**	**like**	to	eat	burgers.	I	really	like	them
\n	\n	I	really	**like**	**to**	eat	burgers.	I	really	like	them

[1] The specific approach to Markov text generation presented here is based on the program presented in *The Practice of Programming* (Kernighan and Pike, Addison-Wesley 1999).

At this point, the table will have the following entries:

Key (n-gram prefix)	**Value (next word)**
\n, \n	I
\n, I	really
I, really	like
really, like	to
like, to	eat

The complete table of all bi-grams in the sample is:

Key (n-gram prefix)	**Value (next word)**
\n, \n	I
\n, I	really
I, really	like
really, like	to, them
like, to	eat
to, eat	burgers.
eat, burgers.	I
burgers., I	really
like, them	with
them, with	bacon., lettuce
with, bacon.	I
bacon., I	eat
I, eat	burgers, them
eat, burgers	with
burgers, with	cheese
with, cheese,	too.
cheese,, too.	I
too., I	eat
eat, them	with
with, lettuce,	too.
lettuce,, too.	\n

In this sample, there are four bi-grams that appear multiple times, [I, really], [really, like], [them, with], and [I, eat]. For three of those bi-grams, different words appear after each occurrence of the bi-gram.

From such a table, a text can be generated by starting with the initial prefix, then appending a word from the list of possible next words. If there are multiple possible words, one is chosen at random to append to the output. Each time a word is chosen, the prefix is updated by removing the first word and adding the chosen word. The process continues until the end marker is appended to the output, as at the end of the sample, or until a pre-determined number of words are generated.

The table below illustrates the generation process

Current Prefix	**Next Word**	**Current generated text**
`\n, \n`	`I`	`I`
`\n, I`	`really`	`I really`
`I, really`	`like`	`I really like`
`really, like`	`them`	`I really like them`
`like, them,`	`with`	`I really like them with`
`them, with`	`bacon., lettuce`	

At this step, there are two alternative words to append to the text. The process continues until the prefix for the end of the sample is reached. In the given sample, only the non-word (i.e. the end of the text) can follow lettuce, too. The text below is one possible output:

```
I really like them with bacon. I eat them with lettuce, too.
```

Programming Homework

Implement the algorithm as a program that accepts a book-length input given as a command line argument. Produces an imitation of 100-200 words. Use prefixes of 2 or 3 words. Upload your solution to GitHub as a ***private*** repository, to which you add **raylangnola** as a collaborator.