**Algorithm for Text Adventure Editor**

Description:

* Editor is organized with a series of choices
  + Each choice will take you allow the user to:
    - Exit the program – kills the editor
    - Load the default game
      * Default game will be a single node game with the options to “start over” or “quit” the game but not exit the program
    - Load a game file
      * This will allow the user to load game.dat from their directory
    - Save the current game
      * Will allow the user to save whatever version of the game is loaded into the program at that instance
    - Edit or Add a Node
      * This will allow the user to edit the game and add nodes to the game as they see fit
      * 5 nodes in total to edit
        + Story
        + menuA
        + nodeA
        + menuB
        + nodeB
      * If nothing is typed into a node, node retains existing data
    - Play the current game
* Game is organized as a series of nodes(or choices)
  + Each choice will take you to another option of choices
  + Each node will have 5 strings:
    - A story string – explains the situation the gamer is in
    - menuA – explains the first option
    - nodeA – is used if menuA is choosen, takes you to next situation until start over or quit is give
    - menuB – explains the second option
    - nodeB – is used if menuB is choosen, takes you to next situation until start over or quit is given
  + From the story string, the user has two options (nodeA / nodeB)
  + If the user choice is “quit” the initial menu will come back

**Pseudocode:**

**main()**

Runs the main loop

Calls a menu

Sends control to other parts of the program

Handles invalid input from menu

**getMenuChoice()**

prints a menu of user options

returns a menu choice

**playGame()**

plays the game

Keeps going until next node is "quit"

**playNode()**

given the game data and a node,

plays out the node

returns the next node

**getDefaultGame()**

creates a single-node default game

returns that data structure

**editNode()**

given the current game structure...

list all the current node content  (json.dumps() would be an easy

way to do this)

get a node name

if that node exists

copy that node to newNode

otherwise...

create newNode with empty data

use editField() to allow user to edit each node

return the now edited newNode

**editField()**

get a field name

print the field's current value

if the user presses 'enter' immediately

retain the current value

otherwise...

use the new value

**saveGame()**

save the game to a data file

you can preset the file name (eg 'game.dat')

print the current game dictionary in human-readable format

Save the file in JSON format

**loadGame()**

presume there is a data file named 'game.dat' in the current

…directory

open that file

load the data into the game object

return that game object