CS100 Fall 2016

Name _____

CPADS Assignment #3
CPADS and the Quest to Implement the Logic in the Holy Grail
Part 1: We Found a Witch – May We Burn Her?

Students of CPADS – you shall have another task to make yourselves an example in these dark times. Behold! Below are links to the *Witch Scene* from *Monty Python and the Holy Grail*. It is now your sacred task to implement the logic from within that scene. That is your purpose - the Quest to Implement the Logic in the Holy Grail.

YouTube video: https://www.youtube.com/watch?v=k3jt5ibfRzw

Script: http://www.montypython.net/scripts/HG-witchscene.php

Watch the video clip, and then review the script. Sir Bedevere asks the villagers many questions, to which they provide responses. Except for one instance, he does not proceed to the next question, or the next portion of his argument, until he has received correct input from the villagers (or King Arthur, in one case).

Your next task is to implement the logic from the Witch Scene in Python (which, by the way, was named after Monty Python). You have been provided with skeleton code that implements much of the scene. There are several TODO's throughout the code that you must implement in order to get the entire scene completely working.

The TODO's are shown in the code, with a description of what to do for each. As you are about to find out, implementing logical expressions can be quite "interesting" – depending on your definition of interesting. Be forewarned – this is a complicated assignment. It will take a good deal of time to get all of the logic correct. There is no one single correct solution, but there are correct results. We will also introduce you to debugging techniques using the PyCharm IDE while working on this assignment.

Here is the list of the TODOs:

1) Add a **`while`** loop to the **`butYourAreDressedAsOne()`** function. The **`while`** loop should run until the witch enters the phrase, "*they dressed me up like this*". Give the witch 3 attempts to get it right. Return **TRUE** if she gets it correct and **FALSE** otherwise.
2) Add a **`while`** loop to the **`andThisIsntMyNose()`** function. The **`while`** loop should run until the villagers enter one of two phrases: either "*well, we did do the nose*" OR "*that's her real nose*". Do not limit the # of attempts for this **`while`** loop. Return **TRUE** for "*well, we did do the nose*", and **FALSE** otherwise.

Name _____

3) Add a **while** loop to the **`andTheHat()`** function. The loop should run until the villagers enter one of two phrases: "*and the hat*" OR "*she was wearing that hat*". If the phrase "*and the hat*" is entered, output the provided messages, and return **TRUE**. Otherwise, return **FALSE**.
4) Add logic to the **`didYouDressHerUpLikeThis()`** function that implements the villagers reply to Sir Bedevere's question.

   This is a complicated one: if at any point, the villagers say "no" 4 or more times in a row, then return FALSE. If the villagers say "no" exactly 3 times in a row AND then say "yes" AT LEAST 2 times in a row, AND then say "a bit" exactly 2 times in a row, you should return TRUE. If they enter any other sequence, you should start over. This requires a while loop, and quite a bit of logic.
   NOTE: This one takes a lot of thought and experimentation to figure out. Use the provided variables to keep track of the various consecutive word counts. If you are looking for consecutive words, and a word comes in that is out of place, remember to restart the appropriate counts.
   Here are some examples of input, and the expected return value:
   a) no, no, no, no – FALSE
   b) no, no, no, yes, yes, a bit, a bit – TRUE
   c) no, no, no, yes, yes, yes, a bit, a bit - TRUE
   d) any other sequence should take you back to the start of the sequence, so that you start counting no's again.

   Treat it like this: you need 3 consecutive *no*'s to start looking for *yes*'s. If you get 2 or more consecutive *yes*'s immediately following the 3 consecutive *no*'s, then start looking for 2 consecutive "*a bit*"s to immediately follow the *yes*'s. If any word is out of sequence, then you go back to counting *no*'s.
5) Add a **while** loop to the **`soHowDoWeTellWhetherSheIsMadeOfWood()`** function that accepts the phrase, "*if she weighs the same as a duck*" and give the villagers three attempts to get it right. If they enter the correct phrase within 3 attempts, output the provided phrase and return TRUE, otherwise, return FALSE.
6) Add a **while** loop to the **`doesWoodSinkInWater()`** function that accepts one of two phrases: "no, it floats" OR "throw her into the pond". Do not limit the # of attempts the villagers have to get it right. If they enter "no, it floats", then output provided message.
7) Add a **while** loop to the **`getWitchWeight()`** function that collects the witch' s weight from the user. The loop should continue to accept input until the entered value is between 2 and 200 pounds inclusive, for a maximum of 4 attempts. The function should accept and return the weight as an integer. Make sure to declare and initialize your test values. Whenever an invalid weight is entered, print the appropriate message. If a valid weight is not entered within the maximum # of attempts, print the third message, and return a weight of 100 for the witch
8) Supply the logic statement that completes the **`sheWeighsTheSameAsADuck()`** function. If the witch's weight is ± 33% of the duck's weight, then print the appropriate message, and return TRUE. Otherwise print the appropriate message and return FALSE.

Name _____

9) And now for something completely different: we're going to use the 5 logical values returned earlier when Sir Bedevere was asking questions of the witch and the villagers, along with the witch's and the duck's weight to determine whether or not the witch is really a witch.

The 5 variables are:

a) **looksLikeAWitch**: TRUE if she looks like a witch, FALSE otherwise
b) **notHerRealClothes:** TRUE if she says they dressed her up like this, FALSE if those are her real clothes
c) **theyDidTheNose:** TRUE if the villagers put the nose on her, FALSE if it is her real nose
d) **theyDidTheHat:** TRUE if the villagers put the hat on her, FALSE if it is her own hat
e) **theyDressedHerUp:** TRUE if the villagers admit to dressing her up, FALSE otherwise

Here is where it gets interesting – you are to write the logic statements to determine whether or not she is a witch, based on those 5 variables AND on her weight compared to that of a duck. The logic is as follows:

if she does NOT look like a witch, she is NOT a witch
else if those are her real clothes, she IS a witch, burn her
else if the villagers did both the nose AND the hat
if the villagers dressed her up
get the duck's weight, and the witch's weight
if she weighs the same as a duck, she IS a witch, burn her
else she does NOT weigh the same as a duck, she is NOT a witch
else the villagers did NOT admit to dressing her up, she IS a witch, burn her
else if the villagers did NEITHER the nose NOR the hat, she IS a witch, burn her
else they did one or the other but NOT both
get the duck's weight, and the witch's weight
if she weighs the same as a duck, she IS a witch, burn her
else she does not weigh the same as a duck, she is NOT a witch

Make sure to call the appropriate functions for:

getting the duck weight: **`getDuckWeight()`**
getting the witch weight: **`getWitchWeight()`**,
comparing those weights: **sheWeighsTheSameAsADuck()**

Pay attention to the return values for each function, and the parameters that must be passed to **`sheWeighsTheSameAsADuck()`**. If she turns out to be a witch, call **`burnHer()`**, and if she isn't, call **`shesNotAWitch().`**

It's that simple… ☺