CSCI1100 : Humanities Computer Science 1

Homework 4

Alternating Characters & Choose Your Own Adventures

Reading: “Videogames for Humans,” by merritt k

This homework is worth 100 points towards your overall homework grade and is due Thursday, February 28 at 11:59:59 PM. There are 2 parts of the homework; part one is worth 40 points and part two is worth 60 points.

Each part will be coded and submitted in separated files. These are the files that will be turned in for this homework:

```
hw4part1.py
hw4part2.py
README.txt
StoryPath.pdf
```

All parts should be submitted by the deadline or your program will be considered late. **You should submit your work on Submitty even though you will not get any auto-grade credit.** We will be overriding these points.

For this homework, we will be building a basic interactive narrative game using string analysis, branching conditionals, and loops. This will be the first homework where we begin connecting our conversations in prior classes with our technical work, including issues of gender and masculinity in technology, the politics of artifacts and computational systems, and the social construction of technology.

Before beginning this homework, you should familiarize yourself with choose-your-own-adventure style games through the reading, a selection from *Videogames for Humans*. The introduction of the reading talks about Twine, an online platform for creating narrative games, as well as some of the social and political impacts Twine’s accessibility has created. The remainder of the chapter is a reflective playthrough of a semi-famous Twine game, *Depression Quest*.

PART ONE: Alternating Vowels and Consonants

Our program will allow users to enter commands to choose paths in the game. Before we begin the game, we should familiarize ourselves with how to read in words, check their case and qualities, and correct inaccurate entries.

Write a program that reads in a word entered by the user and checks whether:

- the word has at least 8 characters,
- starts with a vowel,
- has alternating vowels and consonants, and
- the consonants are in increasing alphabetical order.

Note that you can check letters for alphabetical order using <. For example:

```
>>> ‘a’ < ‘b’
True
>>> ‘z’ < ‘b’
False
```

Your program should work for words entered upper or lower case, and must use a single function **`is_alternating(word)`** that returns **`True`** if the word has the above pattern, and **`False`** otherwise. **Hint:** *Remember to write a loop that goes through each letter and check for the necessary conditions. Using indexing is important here as you need to compare letters in different positions.*

Here is an example run of this program:

```
Enter a word => eLimiNate
eLimiNate
The word 'eliminate' is alternating
```

```
Enter a word => ageneses
ageneses
The word 'ageneses' is not alternating
```

```
Enter a word => ADAGE
ADAGE
The word 'adage' is not alternating
```

The word ‘eliminate’ has this alternating pattern since we have that: 'l' < 'm' < 'n' < 't'.

The word ‘adage’ is not long enough, ageneses has two consonants that are identical and as a result not in strictly increasing alphabetical ordering.

Here is another hint that will help: Given a letter stored in the variable x, the boolean expression:

```
x in 'aeiou'
```

---

is **`True`** if and only if x is a lowercase vowel.

When you have tested your code, please submit it as **`hw4Part1.py`**.

PART TWO: Choose Your Own Adventure

For the remainder of the homework, you will coding a story similar to the reading about Twine games. Your program will be a “choose-your-own-adventure game” that relies on user input to guide a story of your choosing. The homework also has a written and pseudocode requirement, that will walk your instructors through your design process and some reflections on class conversations as explored through code.

The assignment begins by asking you to compose a short “choose your own adventure” story. **The narrative you write should apply some of the lessons that we’ve discussed in class. Such topics include:**

Epistemology; Ideology; Social Construction; Intersections of Gender and Computing; the Politics of Artifacts; Modularization; Systems of Education; Macroethics vs. Microethics

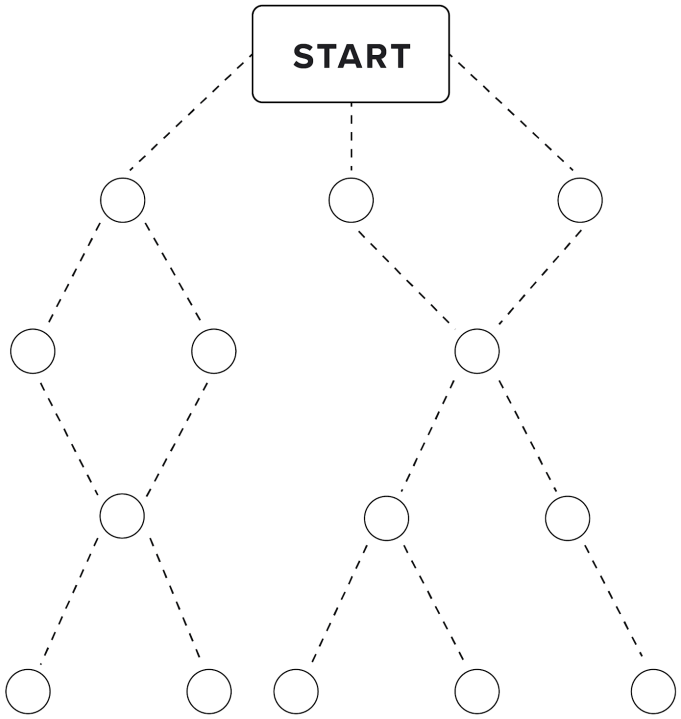
The narrative does not have to be *about* these topics, but should touch on them in some way that shows your understanding of one or more of them as discussed in class and covered in the readings.

We'll be requiring you to turn in more steps than usual for ease of your programming and our understanding of your code and story.

To complete Part 2, you will turn in:

- A PDF diagram of your story (that we can read). This should:
    - Show the overall architecture of your story
    - Label the different conditions needed for each "choice" in the paths
    - If you wrote your story diagram by hand, please **scan** your diagram. Do not take a picture with your phone.
        - There are scanners in the office at the Union and at the Library
- A ReadMe file
    - A **ReadMe.txt** file template is provided. It should contain basic information and your story narrative.
- Your code, titled **`hw4part2.py`**

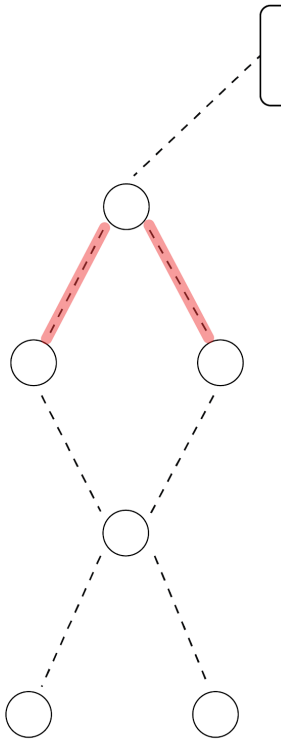
Starting Out



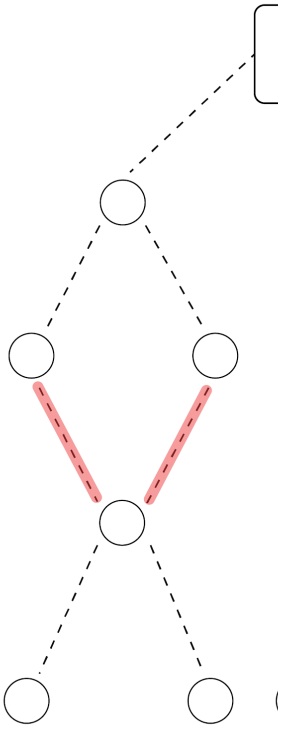
We strongly encourage you to brainstorm some different ideas of what narratives you'd like to explore and think about applying them to a coded narrative. From there, you should start to create a diagram of your story.

To the left is a sample diagram that includes a start, divergences (splits), convergences (merges), and five distinct endings. Below are examples of each highlighted and some more precise definitions.

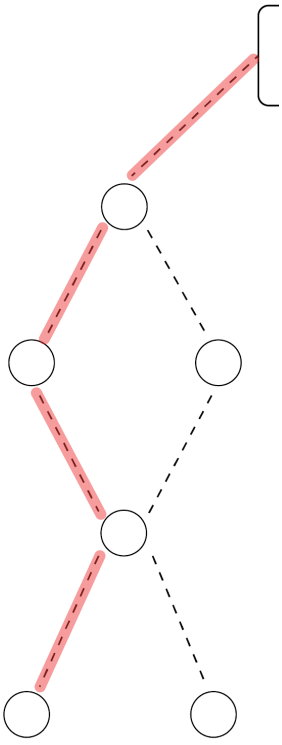
Divergence



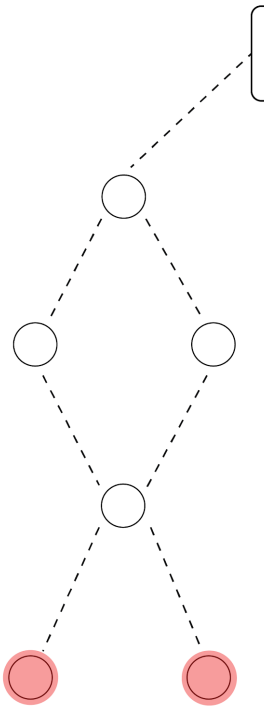
Convergence



Path Length



Endings



A **divergence** is defined as a branching point in the story where the player can make two or more choices that lead to different events.

A **convergence** is defined as a story event that can be reached by more than one path.

Path length is defined as the shortest number of “steps” needed to get from the starting event to an ending event. The diagram above, for example, illustrates a path length of 4.

An **ending event** concludes the story, and gives the players the option to end the story or restart the game.

Your narrative should have at least 4 divergences, 1 convergence, a minimum path length of 3, at least one path length of at least 5, and at least 4 different possible endings.

Once you’ve made your story diagram and checked it against the requirements (ask if you would like a check on that), you can start to transfer this into code. Hold onto this diagram, as you will have to submit it as a part of your homework.

Coding Your Story

Each event your player encounters will present them with narrative text, a choice to be made and/or a convergence point, and a text-based input. Most of the player input will be about making a choice in the story. The appropriate command words that the player could enter should be enclosed in ** . For example:

“I walked up to the door. From what I could tell, the room inside was dark. I could **look in the window down the hallway. I could just take my chances and **open** the door. Or, should I **knock**?”**

In this case, **look, open** and **knock** would be the possible command words a player could enter.

The player input at an ending event will ask either to **end** the game, which will display a list of the choices players made, or **start over**, which will return the player to the starting event.

A key component to this program will be “if” statements. For each divergence in your story will require at least two “if” statements that check for user response, navigate to an event point, or ask the user for new input. If you’re thinking that this is going to get messy, you’re probably right.

There are two requirements that your code has to have.

- **The input your program accepts from the user has to be an exact word or sentenc**e that you check for correctness using an algorithm similar to the first part (consult your code!). This must be executed with a for or a while loop.

    - For example, if your player encounters a door, they may have the option to “open,” “knock,” or “turn around.” Players must enter their commands as text. You will then use a loop to read the string input, convert letter case (so that KNOCK, knock, kNocK, etc. are all “correct” inputs), and either follow the chosen path or return “command not found” and allow the player to rechoose.
    - You can use as many while or for loops as needed throughout your program, but you must use both at least once to complete this check.
- **You have to keep track of your user’s decisions in a list.** At the end of your program, you should print out this list to see what decisions were made. We recommend making this a function and implementing this immediately! **This will help you debugging and is very useful in helping out your code structure *(think hard about this).***

These are some suggestions we have when starting your code to help organize and break down what you’ll be writing:

- Create pseudocode for each divergence and start to match that with each convergence and divergence
- Comment your story into your coding file before you start to code
- **Think of each circle (or a “door”) as a function. This is the most important thing to consider that can make your code easier to organize and debug.** Making functions for each choice may seem more complicated, but this allows you to isolate issues and move things around much easier than just building a large chunk of code. From each function, you can redirect the flow of your program to the other appropriate “doors” to ask the appropriate questions and redirect again to other doors.

Here’s an example of this:

```
def check_case(user_input, choice_given):
      #This function will be the one that checks case sensitiveness
      #In this case, it will return a boolean if the input matches the choice

check_door = input("Do you want to: ‘Go Left’ or ‘Go Right’?”)

#Checking for one decision
if(check_case(check_door, “Go Left”):
      print(“You chose to go left.”)
      #Store the decision in a list.

#Checking for the other decision
if(check_case(check_door, “Go Left”):
      print("You chose to go right.")
      #Store the decision in a list.
```

```
elif:
      #Ask the user again for input until they enter a valid choice

#Use previous decision to move on in your code
```

From here, you should start slowly trying to code your narrative and building it up a bit at a time. Never code all of your homework at once, and especially with this assignment! Either code a path at a time or build out your code from the start and isolate different sections to tackle.

When your user hits an ending event in the story, your program should allow the user to either end the game and print out the list of their choices, or start over from the starting point. Remember to think about what going back to the starting point may do to your list of decisions made.

Turn in your files into Submitty just as you would a regular homework. The autograder will be inaccurate, but Submitty will let you know if you code compiles. We will update your grades in Submitty, so don't worry if your initial submission gives you a very low score. **Remember to ask for help on Piazza from the mentors and your fellow students, and also to attend office hours.**

Grading

Your homework will be graded on:

- Submission of compilable code
- Submission of all required homework components
- Accurate running of the word testing algorithm
- Development of a short story that engages with prior readings and class discussions
- Meeting the minimum story length, ending, divergence, and convergence requirements
- Using at least one "while" loop in your code
- Using at least one "for" loop in your code
- Development of a navigable game
- Passing of "edge case" tests (for example, "kNoCk" as correct input for "knock")
- The ability to end the game at an endpoint and see a list of player decisions made
- The ability to restart the game from an endpoint