CMSC 201 Fall 2018
Homework 2 – Decisions

Assignment: Homework 2 – Decisions
Due Date: Saturday, September 22nd, 2018 by 8:59:59 PM
Value: 40 points

Collaboration: For Homework 2, **collaboration is not allowed** – you must work individually. You may still come to office hours for help, but you may not work with any other CMSC 201 students.

Make sure that you have a complete file header comment at the top of each file, and that all of the information is correctly filled out.

```
# File:    FILENAME.py
# Author:  YOUR NAME
# Date:    THE DATE
# Section: YOUR DISCUSSION SECTION NUMBER
# E-mail:  YOUR_EMAIL@umbc.edu
# Description:
#    DESCRIPTION OF WHAT THE PROGRAM DOES
```

Instructions

For each of the questions below, you are given a problem that you must solve or a task you must complete. For this exercise, you will need to use concepts previously practiced in Homework 1, as well as concepts covered in class during the last week.

You should already be familiar with variables, expressions, `input()`, casting to an integer, and `print()`. You will also need to use one-way and two-way decision structures, as well as nested decision structures. You may also need to use multi-way decision structures for this assignment.

Think carefully about what the overall goal of the algorithm is before you begin coding.

At the end, your Homework 2 files must run without any errors.

NOTE: Your filenames for this homework must match the given ones exactly.

And remember, filenames are case sensitive!

Additional Instructions – Creating the hw2 Directory

During the semester, you'll want to keep your different Python programs organized, organizing them in appropriately named folders (also known as directories).

Just as you did for Homework 1, you should create a directory to store your Homework 2 files. We recommend calling it **`hw2`**, and creating it inside the **`Homeworks`** directory inside the **`201`** directory.

If you need help on how to do this, refer back to the detailed instructions in Homework 1. (You don't need to make a separate folder for each file. You should store all of the Homework 2 files in the same **`hw2`** folder.)

Coding Standards

Prior to this assignment, you should re-read the Coding Standards, linked on the course website at the top of the “Assignments” page.

For now, you should pay special attention to the sections about:

- Naming Conventions
- Use of Whitespace
- Comments (specifically, File Header Comments)
- Line Length

Additional Specifications

For this assignment, **you must use `main()`** as as discussed in class.

For this assignment, you do not need to worry about any “input validation.”

If the user enters a different type of data than what you asked for, your program may crash. This is acceptable.

If the user enters “bogus” data (for example: a negative value when asked for a positive number), your program does not need to worry about correcting the value or fixing it in any way.

For most of the problems below, we do ask you to handle “bogus” data gracefully by displaying something like “invalid option.”

For example, if your program asks the user to enter a whole number, it is acceptable if your program crashes if they enter something else like “dog” or “twenty” or “88.2” instead.

Here is what that might look like:

```
Please enter a number: twenty
Traceback (most recent call last):
  File "test_file.py", line 10, in <module>
    num = int(input("Please enter a number: "))
ValueError: invalid literal for int() with base 10: 'twenty'
```

Questions

Each question is worth the indicated number of points. Following the coding standards is worth 4 points. If you do not have complete file headers and correctly named files, you will lose points.

hw2_part1.py **(Worth 8 points)**

This program simulates an alchemist mixing together different elements to see what happens. You have three different elements to mix together: earth, fire, and water.

Depending on the elements they provide, print out one of three responses:

- If they entered the same element twice:
  - Print out That just makes more <ELEMENT> than you need!
    (where <ELEMENT> is the element they entered)

- If they entered two known elements (earth, fire, or water):
  - Print out <ELEMENT1> and <ELEMENT2> combined make <MIXED>
    (where <MIXED> is the secondary element created)
    - Fire and water creates steam
    - Fire and earth creates lava
    - Water and earth creates mud

- If they enter anything else:
  - Print out <ELEMENT1> and <ELEMENT2> didn't combine.

Your program only needs to work with lowercase letters; you do not need to worry about handling capitalization.

*(HINT: Do **not** start coding this part without having a plan! The order the elements are entered in should not make a difference, so think carefully about what your conditionals should be and if/how they should be nested.)*

(See the next page for sample output.)

Here is some sample output for **hw2_part1.py**, with the user input in **blue**. (Yours does not have to match this word for word, but it should be similar.)

```
linux3[18]% python3 hw2_part1.py
Enter your first choice  (earth, fire, water): earth
Enter your second choice (earth, fire, water): fire
earth and fire combined make lava.

linux3[19]% python3 hw2_part1.py
Enter your first choice  (earth, fire, water): water
Enter your second choice (earth, fire, water): fire
water and fire combined make steam.

linux3[20]% python3 hw2_part1.py
Enter your first choice  (earth, fire, water): earth
Enter your second choice (earth, fire, water): earth
That just makes more earth than you need!

linux3[21]% python3 hw2_part1.py
Enter your first choice  (earth, fire, water): earth
Enter your second choice (earth, fire, water): water
earth and water combined make mud.

linux3[22]% python3 hw2_part1.py
Enter your first choice  (earth, fire, water): water
Enter your second choice (earth, fire, water): earth
water and earth combined make mud.

linux3[23]% python3 hw2_part1.py
Enter your first choice  (earth, fire, water): fire
Enter your second choice (earth, fire, water): water
fire and water combined make steam.
```

hw2_part2.py **(Worth 10 points)**

This program will calculate how many seeds you can buy for your next fall crop in the game Stardew Valley.

*(**WARNING**: This part of the homework is most challenging, so budget plenty of time and brain power. Read the instructions carefully!)*

As a farmer in the game, you have a certain integer amount of gold you can use to buy seeds. Your program will limit your choice of seeds depending on the amount of gold that you have. After you select the single crop that you want to plant, the program will calculate and print how many seeds you can buy and how much gold you would have leftover. Here's the breakdown:

If you have 5000 gold or more you can buy ***cranberry***, ***corn***, or ***sunflower*** seeds.

If you have between 1000 and 4999 gold you can buy ***yams*** or ***pumpkins***.

If you have less than 1000 gold, you should buy ***beets***.

If you enter an amount of gold that is 19 or less, you cannot buy any seeds.

Once you select a seed, you should compute how many seeds you can buy with the amount of gold you have. ***You cannot buy fractional seeds, you can only buy seeds in whole amounts.*** You should then print out how many seeds you can buy, how much they cost in total, and how much gold you will have leftover.

If the user enters a crop name that is unrecognized, the program should say that an invalid option was entered. It should completely skip the calculation of total seed cost, as no valid seed was entered!

Each crop costs the amount of gold listed in the following table:

Cranberry:	240g	**Yam:**	60g
Pumpkin:	100g	**Sunflower:**	200g
Beet:	20g	**Corn:**	150g

The following is some sample output for this program:

```
linux3[14]% python3 hw2_part2.py
How much gold do you have? 5642
Enter your choice (cranberry, sunflower, corn): sunflower
You can buy 28 sunflower seeds for 5600 gold leaving you
with 42 gold.

linux3[15]% python3 hw2_part2.py
How much gold do you have? 2306
Enter your choice (yam, pumpkin): yam
You can buy 38 yam seeds for 2280 gold leaving you with
26 gold.

linux3[16]% python3 hw2_part2.py
How much gold do you have? 15
15 is not enough gold buy any seeds!

linux3[17]% python3 hw2_part2.py
How much gold do you have? 343
You can buy 17 beet seeds for 340 gold leaving you with 3
gold.

linux3[18]% python3 hw2_part2.py
How much gold do you have? 7740
Enter your choice (cranberry, sunflower, corn): corn
You can buy 51 corn seeds for 7650 gold leaving you with
90 gold.

linux3[19]% python3 hw2_part2.py
How much gold do you have? 1000
Enter your choice (yam, pumpkin): pumpkin
You can buy 10 pumpkin seeds for 1000 gold leaving you
with 0 gold.

linux3[20]% python3 hw2_part2.py
How much gold do you have? 10000
Enter your choice (cranberry, sunflower, corn): beet
beet is an invalid option.
```

(more sample output on the next page)

```
linux3[21]% python3 hw2_part2.py
How much gold do you have? 1400
Enter your choice (yam, pumpkin): peppers
peppers is an invalid option.

linux3[22]% python3 hw2_part2.py
How much gold do you have? 5000
Enter your choice (cranberry, sunflower, corn): sunflower
You can buy 25 sunflower seeds for 5000 gold leaving you
with 0 gold.

linux3[23]% python3 hw2_part2.py
How much gold do you have? -1500
-1500 is not enough gold buy any seeds!

linux3[24]% python3 hw2_part2.py
How much gold do you have? 5001
Enter your choice (cranberry, sunflower, corn): cranberry
You can buy 20 cranberry seeds for 4800 gold leaving you
with 201 gold.

linux3[25]% python3 hw2_part2.py
How much gold do you have? 4999
Enter your choice (yam, pumpkin): yam
You can buy 83 yam seeds for 4980 gold leaving you with
19 gold.

linux3[27]% python3 hw2_part2.py
How much gold do you have? 999
You can buy 49 beet seeds for 980 gold leaving you with
19 gold.
```

*(HINT: Do **not** start coding this part without having a plan! If you are stuck, try drawing a flowchart of the different options, or come to office hours for help.)*

hw2_part3.py (Worth 9 points)

This program is a simple symptom checker that is way better than WebMD! It will ask the user about their symptoms and attempts to guess the illness they have based on their answers. There are only six possible illnesses.

*(**WARNING**: This part of the homework is also challenging, so budget plenty of time and brain power. Read the instructions carefully!)*

The program can ask the player about five characteristics. It should ask the ***minimum*** number of questions needed to guess the illness. *(HINT: It should need to ask no less than two questions and no more than three questions to find the right illness.)*

- Do you have a runny nose?
- Do you have a fever?
- Do you have a headache?
- Do you have a cough?
- Do you have red bumps on your skin?

For these inputs, the program can assume the following:

- The user will only ever enter either lowercase **`y`** (for “yes”) or lowercase **`n`** (for “no”)

Based on the user’s responses, the program must select the correct illness and print it to the screen. Here are the possibilities for the illnesses:

- If you don’t have a fever and you have runny nose, you have allergies.
- If you don’t have a fever and you don’t have runny nose, then you’re not sick.
- If you have a fever and headache and bumps, you have the chicken pox.
- If you have a fever and headache and no bumps, you have the common cold.
- If you have a fever and no headache and cough, you have pneumonia.
- If you have a fever and no headache and no cough, you have a sinus infection

*(HINT: Do **not** start coding this part without having a plan! If you are stuck, try drawing a flowchart of the different options, or come to office hours for help.)*

Here is some sample output for **hw2_part3.py**, with the user input in **blue**. (Yours does not have to match this word for word, but it should be similar.)

```
linux3[351]% python3 hw2_part3.py
Do you have a fever? (y/n): n
Do you have a runny nose? (y/n): n
I don't think you are sick.

linux3[352]% python3 hw2_part3.py
Do you have a fever? (y/n): n
Do you have a runny nose? (y/n): y
You must have allergies.

linux3[353]% python3 hw2_part3.py
Do you have a fever? (y/n): y
Do you have a headache? (y/n): y
Do you have red bumps on your skin? (y/n): y
You must have the chicken pox.

linux3[354]% python3 hw2_part3.py
Do you have a fever? (y/n): y
Do you have a headache? (y/n): y
Do you have red bumps on your skin? (y/n): n
You must have the common cold.

linux3[355]% python3 hw2_part3.py
Do you have a fever? (y/n): y
Do you have a headache? (y/n): n
Do you have a cough? (y/n): y
You must have pneumonia.

linux3[356]% python3 hw2_part3.py
Do you have a fever? (y/n): y
Do you have a headache? (y/n): n
Do you have a cough? (y/n): n
You must have a sinus infection.
```

hw2_part4.py **(Worth 5 points)**

For this program, create a (very simplified) day of the week calculator. Ask the user to enter the day of the month, and respond with the correct day of the week.

The program will assume that the month starts on Saturday and has 30 days (just like the month of September in 2018). The program

- Can assume that the number entered will be an integer
- Cannot assume that the number entered will be valid!

If the day of the month the user entered is not a valid day of the month (less than 1 or greater than 30), simply print a short error message to the user. Otherwise, print the day of the week that day falls on. For instance, the 2nd would be a Sunday, the 10th would be a Monday, etc.

IMPORTANT: Do not write a case for each day of the month. If your program uses dozens of individual **`if`**, **`elif`**, or **`else`** statements, you will lose significant points.

(HINT: There is a mathematical operator in Python that will allow you to write this program without needing to have dozens of individual decision statements. Review Lecture 03 (Operators) to see it in action.)

(See the next page for sample output.)

Here is some sample output for **hw2_part4.py**, with the user input in **blue**. (Yours does not have to match this word for word, but it should be similar.)

```
linux3[87]% python3 hw2_part4.py
Enter the day of the month: 35
35 is an invalid day in September.

linux3[88]% python3 hw2_part4.py
Enter the day of the month: 28
September 28 is a Friday.

linux3[89]% python3 hw2_part4.py
Enter the day of the month: 22
September 22 is a Saturday.

linux3[90]% python3 hw2_part4.py
Enter the day of the month: 3
September 3 is a Monday.

linux3[91]% python3 hw2_part4.py
Enter the day of the month: 13
September 13 is a Thursday

linux3[92]% python3 hw2_part4.py
Enter the day of the month: 10
September 10 is a Monday.
```

hw2_part5.py **(Worth 4 points)**

For this last program, you are going to ask the user about the state of three switches, and then use this information to determine the state of a lightbulb.

For the input to these two questions, you can assume the following:

- The user will only ever enter either lowercase **`y`** (for "yes") or lowercase **`n`** (for "no")

The lightbulb is only on if both the following statements hold true:

- the first switch is off, or the second switch is on
- the third switch is off.

The lightbulb is off in all other combinations of switches.

IMPORTANT: You can only use a single **`if-else`** statement within this program! Think carefully about how you can accomplish this.
(The **`if`** statement can have a combination of multiple **`and`** and **`or`** statements, if you think that is needed to solve the problem.)

Here is some sample output, with the user input in **blue**. This sample output only shows you 4 of the 8 possible combinations of switches—you should test the other 4 yourself to make sure they each follow the rules above.

(Yours does not have to match this word for word, but it should be similar.)

```
linux3[40]% python3 hw2_part5.py
Please enter 'y' for yes and 'n' for no.
Is the first switch on? (y/n): y
Is the second switch on? (y/n): y
Is the third switch on? (y/n): y
The light is off.

linux3[41]% python3 hw2_part5.py
Please enter 'y' for yes and 'n' for no.
Is the first switch on? (y/n): n
Is the second switch on? (y/n): y
Is the third switch on? (y/n): n
The light is on.
```

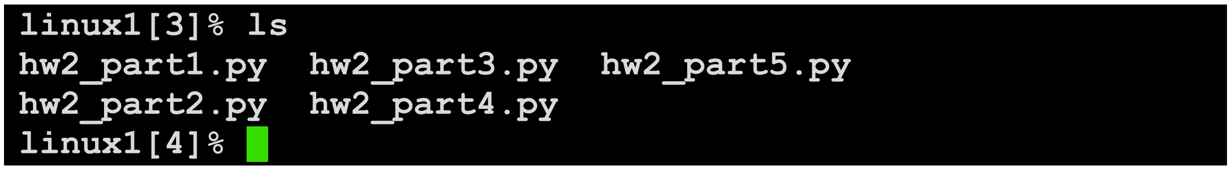
```
linux3[42]% python3 hw2_part5.py
Please enter 'y' for yes and 'n' for no.
Is the first switch on? (y/n): y
Is the second switch on? (y/n): y
Is the third switch on? (y/n): n
The light is on.

linux3[43]% python3 hw2_part6.py
Please enter 'y' for yes and 'n' for no.
Is the first switch on? (y/n): n
Is the second switch on? (y/n): y
Is the third switch on? (y/n): y
The light is off.
```

Submitting

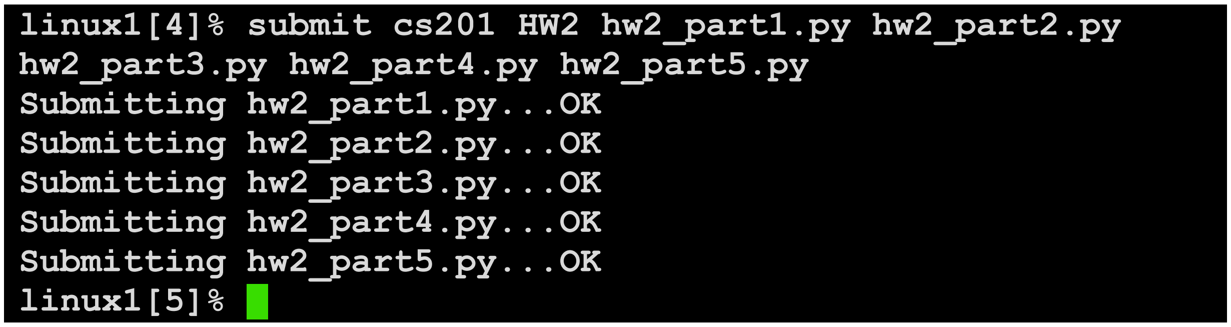
Once your **`hw2_part1.py`**, **`hw2_part2.py`**, **`hw2_part3.py`**, **`hw2_part4.py`**, and **`hw2_part5.py`** files are complete, it is time to turn them in with the **`submit`** command. (You may also turn in individual files as you complete them. To do so, only **`submit`** those files that are complete.)

You must be logged into your account on GL, and you must be in the same directory as your Homework 2 Python files. To double-check you are in the directory with the correct files, you can type **`ls`**.



```
linux1[3]% ls
hw2_part1.py  hw2_part3.py  hw2_part5.py
hw2_part2.py  hw2_part4.py
linux1[4]% 
```

To submit your Homework 2 Python files, we use the **`submit`** command, where the class is **`cs201`**, and the assignment is **`HW2`**. Type in (all on one line) **`submit cs201 HW2 hw2_part1.py hw2_part2.py hw2_part3.py hw2_part4.py hw2_part5.py hw2_part6.py`** and press enter.



```
linux1[4]% submit cs201 HW2 hw2_part1.py hw2_part2.py
hw2_part3.py hw2_part4.py hw2_part5.py
Submitting hw2_part1.py...OK
Submitting hw2_part2.py...OK
Submitting hw2_part3.py...OK
Submitting hw2_part4.py...OK
Submitting hw2_part5.py...OK
linux1[5]% 
```

If you don't get a confirmation like the one above, check that you have not made any typos or errors in the command.

You can check that your homework was submitted by following the directions in Homework 0. Double-check that you submitted your homework correctly, since **an empty file will result in a grade of zero for this assignment.**