CMSC 201 Spring 2017
Homework 2 – Decisions

Assignment: Homework 2 – Decisions
Due Date: Friday, February 17th, 2017 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 four Homework 2 files in the same **`hw2`** folder.)

Objective

Homework 2 is designed to help you practice decision structures in a Python environment. You will 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.

Remember to enable Python 3 before running and testing your code:

```
scl enable python33 bash
```

Task

For all four of the programs, you will need to interact with the user by receiving input from them. Some of the program will also require that you use "new" mathematical operators like modulus and integer division.

For all four of the programs, you will need to use at least one decision structure.

Specification

Prior to this assignment, you should re-read the Coding Standards, available on Blackboard under "Assignments" and 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)

You should start forming good habits now. Make sure to pay attention to your TA's feedback when you receive your Homework 2 grade back.

Additional Specifications

For this assignment, **you must use** `main()` as seen in your **`lab2.py`** file, and 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 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, having complete file headers, and having correctly named files is worth 6 points.

hw2_part1.py **(Worth 7 points)**

Ask the user for two inputs (in this **exact** order): the name of the class, and the letter grade they earned. Then, depending on their letter grade, print out the outcome, making sure to restate the name of the class in the output.

- If they earned an "A", a "B", a "C", or a "D" they **passed**
- If they earned an "F" they **failed**
- If they earned a "W" they **withdrew**
- If they earned any other score not a valid grade

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

```
bash-4.1$ python hw2_part1.py
Please enter the name of the class: Extra-Quantum Physics
Please enter your letter grade: D
You passed Extra-Quantum Physics

bash-4.1$ python hw2_part1.py
Please enter the name of the class: DOGS 101
Please enter your letter grade: A+
A+ is not a valid grade for DOGS 101

bash-4.1$ python hw2_part1.py
Please enter the name of the class: Underwater Basketball
Please enter your letter grade: W
You withdrew from Underwater Basketball

bash-4.1$ python hw2_part1.py
Please enter the name of the class: DIFF 658
Please enter your letter grade: F
You failed DIFF 658
```

hw2_part2.py **(Worth 6 points)**

Ask the user to enter a number of seconds (read in as an integer). Calculate and print out how many whole minutes, hours, or days are contained in the number of seconds specified by the user.
If the number input is negative, print out an error message instead.

As a reminder, there are

- 60 seconds in a whole minute
- 3,600 seconds in a whole hour
- 86,400 seconds in a whole day

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

```
bash-4.1$ python hw2_part2.py
Please enter a number of seconds: -85
Cannot calculate for a negative number

bash-4.1$ python hw2_part2.py
Please enter a number of seconds: 59
There are 0 minutes in 59 seconds
There are 0 hours in 59 seconds
There are 0 days in 59 seconds

bash-4.1$ python hw2_part2.py
Please enter a number of seconds: 60
There are 1 minutes in 60 seconds
There are 0 hours in 60 seconds
There are 0 days in 60 seconds

bash-4.1$ python hw2_part2.py
Please enter a number of seconds: 90000
There are 1500 minutes in 90000 seconds
There are 25 hours in 90000 seconds
There are 1 days in 90000 seconds
```

hw2_part3.py **(Worth 14 points)**

This program plays a simple game, where it asks the player about their dog, and attempts to guess the dog's breed based on their answers. For simplicity's sake, there are only five possible dog breeds.

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

- Does the dog have short legs?
- Does the dog have long fur?
- Is the dog a big dog?
- Does the dog have a curly tail?

For these inputs, the program can assume the following:

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

Based on the user's responses, the program must select the correct breed and print it to the screen. Here are the possibilities for the dog breeds:

- Dog has short legs and long fur: Shetland Sheepdog
- Dog has short legs and does *not* have long fur: Swedish Vallhund
- Dog does *not* have short legs and is *not* a big dog: Azawakh
- Dog does *not* have short legs, is a big dog, and has a curly tail: Kangal
- Dog does *not* have short legs, is a big dog, and *does* not have a curly tail: Irish Wolfhound

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

*(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.)*

(See the next page for sample output.)

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

Make sure you spell all of the dog breeds correctly!

```
bash-4.1$ python hw2_part3.py
Please enter 'yes' or 'no' to these questions.
Does your dog have short legs? yes
Does your dog have long fur? no
Your dog is a Swedish Vallhund!

bash-4.1$ python hw2_part3.py
Please enter 'yes' or 'no' to these questions.
Does your dog have short legs? no
Is your dog a big dog? yes
Does your dog have a curly tail? yes
Your dog is a Kangal!

bash-4.1$ python hw2_part3.py
Please enter 'yes' or 'no' to these questions.
Does your dog have short legs? no
Is your dog a big dog? no
Your dog is an Azawakh!

bash-4.1$ python hw2_part3.py
Please enter 'yes' or 'no' to these questions.
Does your dog have short legs? no
Is your dog a big dog? yes
Does your dog have a curly tail? no
Your dog is an Irish Wolfhound!

bash-4.1$ python hw2_part3.py
Please enter 'yes' or 'no' to these questions.
Does your dog have short legs? yes
Does your dog have long fur? yes
Your dog is a Shetland Sheepdog!
```

hw2_part4.py **(Worth 7 points)**

Finally, 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 Wednesday and has 28 days (just like the month of February, 2017). 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 28), 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 Thursday, the 10th would be a Friday, etc.

IMPORTANT: Do not write a case for each day of the month. If your program uses 20 (or more) 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 over 20 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.)

```
bash-4.1$ python hw2_part4.py
Please enter the day of the month: 17
Today is a Friday!

bash-4.1$ python hw2_part4.py
Please enter the day of the month: 29
The date 29 is an invalid day.

bash-4.1$ python hw2_part4.py
Please enter the day of the month: 7
Today is a Tuesday!

bash-4.1$ python hw2_part4.py
Please enter the day of the month: 0
The date 0 is an invalid day.

bash-4.1$ python hw2_part4.py
Please enter the day of the month: 1
Today is a Wednesday!
```

Submitting

Once your **`hw2_part1.py`**, **`hw2_part2.py`**, **`hw2_part3.py`**, and **`hw2_part4.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 1 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_part2.py  hw2_part3.py  hw2_part4.py
linux1[4]%
```

To submit your Homework 1 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`** and press enter.

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