CSCI 1133, Fall 2019
Programming Assignment 3
Due: 11:55pm, Wednesday September 25, 2019

Due Date: Submit your solutions to GitHub by 11:55 p.m., Wednesday, September 25th. We will do a pull from this time point. Do not upload anything to Canvas and PLEASE be sure to use proper naming conventions for the file, classes, and functions. We will NOT change anything to run it using our scripts.

Unlike the computer lab exercises, this is not a collaborative assignment. You must design, implement, and test your code on your own without the assistance of anyone other than the course instructor or TAs. In addition, you may not include solutions or portions of solutions obtained from any source other than those provided in class (so you are ONLY allowed to reuse examples from the textbook, lectures, or code you and your partner write to solve lab problems). Otherwise obtaining or providing solutions to any homework problem for this class is considered Academic Misconduct. See the syllabus and read section "Academic Dishonesty" for information concerning cheating. Always feel free to ask the instructor or the TAs if you are unsure of something. They will be more than glad to answer any questions that you have. We want you to be successful and learn so give us the chance to help you.

Instructions: This assignment consists of 3 problems, worth a total of 40 points. Solve the problems below by yourself, and put all functions in a single file called `hw3.py`. Use the signatures given for each class and function. We will be calling your functions with our test cases so you must use the information provided. If you have questions, ask!

Because homework files are submitted and tested electronically, the following are very important:

- You follow all naming conventions mentioned in this homework description.
- You submit the correct file, hw3.py, through Github by the due date deadline.
- You follow the example input and output formats shown.
- Regardless of how or where you develop your solutions, your programs should execute using the python3 command on CSELabs computers running the Linux operating system.

Push your work into Github under your own repo. The specific hosting directory should be: repo-<username>/hw3, where you replace <username> with your U of M user name. For instance, if your email address is bondx007@umn.edu, you should push your hw3 to this directory: repo-bondx007/hw3, meaning that the path to your file is `repo-bondx007/hw3/hw3.py`

The following will result in a score reduction equal to a percentage of the total possible points:

- Incorrectly named/submitted source file, functions, or classes (20%)
- Constraints not followed (40%)
- Failure to execute due to syntax errors (30%)
- Not filling out the documentation template for every function, other bad code style (10%)

Use the following template for EVERY function that you write (note: a helper function is a function so it gets a template.)

```
#==========================================
# Purpose: (What does the function do?)
# Input Parameter(s): (Each parameter by name and what it represents)
# Return Value(s): (What gets returned? Possibilities?)
#==========================================
```

Problem A. (10 *points*) **Bark Scale**

Write a function `sound(weight)` that takes in a single positive integer value `weight`, representing the weight of a given dog, rounded to the nearest pound. The function returns a string representing what the dog's bark would probably sound like, based on the following scale:

- Less than 13 lbs: 'Yip'
- 13 - 30 lbs: 'Ruff'
- 31 - 70 lbs: 'Bark'
- More than 70 lbs: 'Boof'

Constraints:

- Do not import/use any Python modules.
- Do not use the input() function,
- Your submission should have no code outside of function definitions (comments are fine)

Examples:

```
>>> sound(13)
'Ruff'

>>> sound(12)
'Yip'

>>> sound(50)
'Bark'

>>> sound(10000)
'Boof'
```

Problem B. (10 *points*) **Text Adventure Choice**

Some of the earliest computer games developed were Interactive Fiction games, in which the user's environment is described in text, and the user makes choices using text commands. In this problem and the next one, we'll be developing a very simple text-based adventure game. Every choice in this game will have exactly three options, so we can write a function that works for any of them.

Write a function `choice(text,option1,option2,option3)`, that takes in four string values. `text` is a string representing the prompt for a choice in a text adventure game, and `option1`, `option2`, and `option3` are strings representing the three possible options. The function should print out the text, and then print out the options (label them with 1., 2., and 3.). Next, the input() function should be used to prompt the user to choose 1, 2, or 3; if the user does not choose one of those options, then the function must print "Invalid option" and prompt the user again (use a while loop for this). Finally, the function should return the one character string that represents the user's choice: `'1'`, `'2'`, or `'3'`. See the examples below for an idea of how to format the printing and input functions: you don't have to match it exactly but it should be roughly the same.

Constraints:

- Do not import/use any Python modules.
- Your submission should have no code outside of function definitions (comments are fine)

Examples (text in **bold** is returned, text in red is user input, text in *italics* is printed):

```
>>> choice("You have no idea how to approach this problem.",
       "Go to office hours",
       "Send an email to the TAs",
       "Post the problem online, there's no way I'll be caught")
You have no idea how to approach this problem.

1. Go to office hours
2. Send an email to the TAs
3. Post the problem online, there's no way I'll be caught
Choose 1, 2, or 3: 1
'1'

>>> choice("You must decide the fate of the galaxy.",
       "Destroy the robots",
       "Control the robots",
       "Merge with the robots?")
You must decide the fate of the galaxy.

1. Destroy the robots
2. Control the robots
3. Merge with the robots?
```

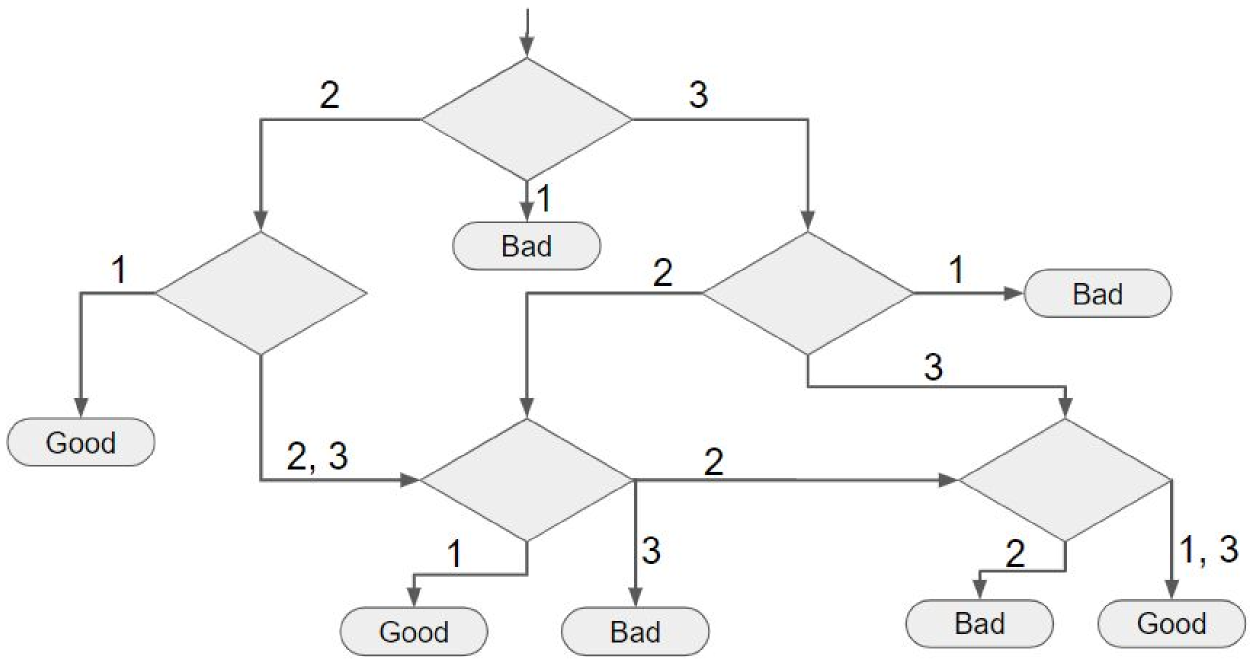
```
Choose 1, 2, or 3: 4
Invalid option
Choose 1, 2, or 3: Sing at the robots
Invalid option
Choose 1, 2, or 3: -2
Invalid option
Choose 1, 2, or 3: 2
'2'

>>> choice("Choose a house.",
       "Black Eagles",
       "Slytherin",
       "Team Instinct")
Choose a house.

1. Black Eagles
2. Slytherin
3. Team Instinct
Choose 1, 2, or 3: Black Eagles
Invalid option
Choose 1, 2, or 3: 3
'3'
```

Problem C. (20 *points*) **Text Adventure Game**

Using the function from the previous problem, write a function `adventure()` which takes in no arguments, and gives the user a sequence of choices, leading to one of several endings. Since we want this problem to not be impossible to grade, you need to follow a very specific choice structure, namely the one outlined in the flowchart below:

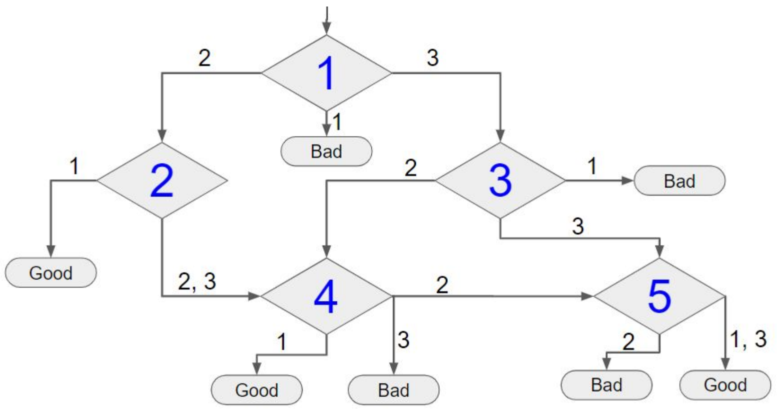


Each diamond represents a decision, for which you must use your `choice` function from problem B. Each arrow represents one of the three possible choices for that decision: 1, 2, or 3. Each rounded box represents an ending to the story, which comes in one of two possible types: Good (the user succeeds in some way) or Bad (the user fails). The story itself is entirely up to you, with one exception: you can't use the one in the example below. We will not be grading you based on the story's quality, logic, originality, or grammatical correctness, so if you're lazy and just want to put in empty strings for every single choice, that's fine, but **you must follow the choice structure in the flowchart above**.

Your function should **return** the boolean value (not the string) `True` if the user reached a Good ending, or `False` if they reached a Bad ending. You can put in a few additional print statements beyond the ones inside the `choice` function to better explain the story, but this is not required.

Hints:

- There are two basic ways to do this. You could put in nested if statements for every possible combination of decisions, which will result in a lot of repeated code, since there are multiple paths to two of the decision points. Or, you could treat this as a state machine. In particular, if we label each decision as shown below, you'll notice that every path through the flowchart passes through states (decisions) in an increasing order (a path may skip some states, but it will never go from a higher numbered state, down to a lower numbered one).



- You can take advantage of the above by using an extra variable to track what state the user is currently in (what number decision point they are currently at), and changing it every time they make a choice. For each state starting at 1 and ending at 5, check whether the user is in that state, and only give them the respective choice if they are. Since all paths go through states in an increasing numerical order, you only have to check each state once.

Constraints:
- Do not import/use any Python modules
- Your submission should have no code outside of function definitions (comments are fine).
- You must use the `choice()` function from problem B.

Examples (text in **bold** is returned, text in red is user input, text in *italics* is printed. Note that because you are required to have a different story than mine, the printed text should not match, but the same sequence of inputs should lead to the same return value):

```
>>> adventure()
You sneak into the dragon's lair, with your comrades Wizard
McBlastyFace and Steve the Bard.  The dragon is fast asleep.

1. Tickle the dragon on the nose
2. Tell your team to start stealing things
3. Tell your team to prepare for battle
Choose 1, 2, or 3: 1

The dragon sneezes out a fireball and incinerates you instantly.
False

>>> adventure()
You sneak into the dragon's lair, with your comrades Wizard
McBlastyFace and Steve the Bard.  The dragon is fast asleep.

1. Tickle the dragon on the nose
2. Tell your team to start stealing things
```

```
3. Tell your team to prepare for battle
Choose 1, 2, or 3: 4
Invalid option
Choose 1, 2, or 3: 2

You can't carry the entire hoard of loot.

1. Take the pile of silk in the corner
2. Take as much gold as you can carry
3. Take the huge diamond nestled under the dragon's claw
Choose 1, 2, or 3: 1

You and your team easily take all of the silk and escape quietly into
the night.
True

>>> adventure()
You sneak into the dragon's lair, with your comrades Wizard
McBlastyFace and Steve the Bard.  The dragon is fast asleep.

1. Tickle the dragon on the nose
2. Tell your team to start stealing things
3. Tell your team to prepare for battle
Choose 1, 2, or 3: 3

Who will lead the attack?

1. Wizard McBlastyFace
2. Steve the Bard
3. Me, of course
Choose 1, 2, or 3: 2

Steve trips over a rock and the dragon wakes up.

1. Tell Steve to sing to the dragon
2. CHARGE!
3. Run away
Choose 1, 2, or 3: 2

You surprise the dragon, wounding it greatly.  But now the dragon
turns to you and prepares to scorch you with its fire breath.

1. Hide behind a marble pillar
```

```
2. Hide behind your shield
3. Run between its legs
Choose 1, 2, or 3: 1

Your cloak is a little singed, but the marble deflects most of the
fire.  Wizard McBlastyFace disintegrates the dragon with a single
spell.
True
```