HW 2: Conditionals

Due Sep 29th, 6pm, codePost.io
Please submit all files created for this assignment to codePost, including the text file. You can submit as many times as you like up until the deadline. ***If you are planning to use one or more late days, please notify us by sending a private Piazza message to "Instructors".***

Familiarize yourself with how to approach a programming assignment and review the grading rubric on this assignment's page in Canvas before getting started.

Code style, documentation, and other requirements
codePost style tests now count toward your homework score.

The formatting guidelines introduced in Labs 1 & 2 are ***requirements*** for all assignments: every file must have a file comment, your programs must start with `main()`, use constants instead of magic numbers and strings. For this assignment, you do not need to include test cases in your file comments.

A portion of your score will be based on the simplicity and readability of your code. Even if your code passes all correctness tests, you will lose points for overly complex or repetitive conditional logic. Use string methods to keep the input validation as simple as you can and avoid redundancy in your conditional branches—don't repeat yourself!

The focus of this homework is conditionals, so there are lots of conditional branches to figure out, particularly in problem 3. If you're struggling to keep all the requirements straight and pass the codePost tests, try drawing flowcharts to represent the branching logic.

Written Component (10 pts)

Save your answers in a plain text file called written.txt.

Question 1: Consider the Python code snippet below. Line numbers are indicated on the left hand side. All of your answers below should begin with line 1. (1pt each)

```
1 choice = input("What would you like for breakfast? ")
2 if choice == "eggs":
3   print("Sorry, we're out of eggs.")
4 elif choice == "toast":
5   print("Coming right up!")
6 print("Thank you for your order.")
```

a) List the line numbers in the order in which they're executed if the user types "toast" at the prompt.
b) List the line numbers in the order in which they're executed if the user types "Eggs" at the prompt.
c) List the line numbers in the order in which they're executed if the user types "eggs" at the prompt.
d) List the line numbers in the order in which they're executed if the user types "bacon" at the prompt.

Question 2: For each Python snippet below, what would be printed to the console? Be sure to preserve line spacing in your answers, if applicable. (1pt each)

```
a) if 6 > 2 * 5:
       print("Hello")
   print("World")

b) if 6 - 2 < 5:
       print("Hello")
       print("World")

c) x = 3
   y = 2
   if x > 2:
       if y > 2:
           z = x + y
           print("z =", z)
       x = x + y
   print("x =", x)
```

Question 3: What do each of the following boolean expressions evaluate to? (0.5pts each)

```
a) 10 % 3 <= 10 // 3
b) 10 % 3 != 10 // 3
c) False and False
d) False or True
e) not True and True
f) "Hello" + " " + "World" == "Hello World"
```

Programming Component (50 pts) + 10 or 15?

Problem 1: registration.py

You have been tasked with prototyping part of a program that will help students at a small college register for classes.

When the program starts, it should prompt the user to enter the course number they want to register for. Use the following message, “`Enter a course number: `”. The user must enter a valid course number to continue. Valid course numbers at the college are X101, X102, B500, B525, and B701. Your program should give the user some flexibility—the letter in the course number can be upper or lower case and they can leave a space between the letter and number. For example, here are all the valid ways to enter course X101: “X101”, “x101”, “X 101”, “x 101”. If the user enters an invalid course number, print the message, “`Invalid course number`”.

X101 and X102 have no prerequisites so, if the user wants to register for either of these courses, print the message, “`You have successfully registered for <course number>`”, where `<course number>` is the selected course in standard format (uppercase letter followed by the digits with no space between).

If the user wants to register for any of the B courses, they must have earned an A or B in X101 and an A, B, or C in X102. The college grading system does not include + or - grades. Prompt the user to enter their grades for the prerequisites using the following prompts:

- “`What grade did you get for X101? `”
- “`What grade did you get for X102? `”

The user should be able to enter lower case or upper case letters e.g. “a” and “A” are equivalent.

If the user meets the prerequisites, print “`You meet all the prerequisites and have successfully registered for <course number>`”. If the user does not meet the prerequisites (or enters invalid grades), print “`You do not meet the prerequisites for <course number>`”. In both messages, `<course number>` is the selected course in standard format (uppercase letter followed by the digits with no space between).

The codePost tests assume the course number is validated as soon as it is entered (see example 2). The tests also assume that, when the user has entered a course number with prerequisites, the grade requirements are not checked until the user has entered their grades for both X101 and X102 (see example 1).

Example input / output # 1 (user input is shown in blue)

```
Enter a course number: b 701
What grade did you get for X101? F
What grade did you get for X102? A
You do not meet the prerequisites for B701
```

Example input / output # 2 (user input is shown in blue)

```
Enter a course number: 102
Invalid course number
```

Problem 2: shapes.py

Write a program that calculates the area of a shape based on information supplied by the user.

First, ask the user to select a shape using the following prompt: "`Select a shape (triangle, square, or rectangle): `". As with problem 1, input should be case-insensitive—accept uppercase, lowercase, and mixed case input. If the user enters an invalid shape, print the message, "`Unknown shape`".

Next, you will need to prompt the user for shape dimensions so you can calculate the area. Different shapes require different dimensions:

- For rectangles and triangles, ask for the width then the height. (For the purposes of this homework, we're using the term "width" instead of "base" for triangles.)
- For squares, only prompt for the width.

The prompts to use:

- "`Enter the width: `"
- "`Enter the height: `"

You can assume that the user will respond with a number (vs. a word or a boolean) but don't assume they will enter a valid dimension. To be valid, a dimension must be greater than 0. If the user enters an invalid dimension, print "`Invalid width`" or "`Invalid height`" as appropriate.

Finally, calculate the shape's area and print the result using the following format:
`The area of the` **`<shape>`** `is` **`<area>`**
...where `<shape>` is the shape selected by the user (all lowercase) and `<area>` is the area of the shape to two decimal places.

The codePost tests assume that each input is validated as soon as it is entered (see example 2).

Example input / output # 1 (user input is shown in blue)

```
Select a shape (triangle, square, or rectangle): rectangle
Enter the width: 3
Enter the height: 4
The area of the rectangle is 12.00
```

Example input / output # 2 (user input is shown in blue)

```
Select a shape (triangle, square, or rectangle): triangle
Enter the width: -35
Invalid width
```

Problem 3: exercise.py

Write a program that will create an exercise plan based on the day of the week and the weather.

Start by prompting the user to enter the following information:

- `"What day is it? "` Accept the following as input (case-insensitive): “M”, “Tu”, “W”, “Th”, “F”, “Sa”, or “Su”.
- `"Is it a holiday? "` Accept “Y” or “N” (case-insensitive) as input for “yes” or “no”.
- `"Is it raining? "` Accept “Y” or “N” (case-insensitive) as input for “yes” or “no”.
- `"What is the temperature? "` Valid input is any number.

You can assume the user enters the correct data *type* at each prompt, but do not assume their input will be valid. If any of the user’s inputs are invalid, the program should output the default workout, `"Swim for 35 minutes",` *after* the user has answered *all* questions.

If all inputs are valid, your program should decide on a workout based on the rules below. Note that some later bullet points override earlier bullet points!

- Workout days are Monday, Wednesday, Friday, Saturday, and holidays. For example, Thanksgiving is a holiday so it would be considered a workout day even though it falls on a Thursday.
- Non-workout days are rest days.
- If it is a Monday, Wednesday, or Friday, go running.
- If it is a Saturday or a holiday, go for a hike (even if the holiday falls on a Monday, Wednesday, or Friday).
- If it is raining and it is a workout day, go swimming instead of the usual activity.
- The duration of the exercise should be 45 minutes unless it’s a running day and the temperature is above 75 degrees or below 35 degrees, in which case, limit the duration to 30 minutes. Other activities besides running are not impacted by the temperature.

Use this format for the output message on workout days:

```
<Exercise> for <duration> minutes
```

...where **`<Exercise>`** is one of “Run”, “Swim”, or “Hike”, and **`<duration>`** is either “45” or “30”.

Use this message for rest days:

```
Take a rest day
```

The codePost tests assume all input questions will be asked, even if some of the answers are not strictly relevant based on the user’s earlier input (see example 1). The codePost tests also assume that inputs are not validated until after the user has answered all questions (see example 2).

Example input / output # 1 (user input is shown in blue)

```
What day is it? m
Is it a holiday? y
Is it raining? n
What is the temperature? 34
Hike for 45 minutes
```

Example input / output # 2 (user input is shown in blue)

```
What day is it? TU
Is it a holiday? N
Is it raining? Kind of
What is the temperature? 85
Swim for 35 minutes
```