COP 2930 - Individual Programming Assignment #5

Due date: Please consult WebCourses for your due date/time

Objectives

1. To use a while loop to solve a problem that uses an unknown number of repetitions.
2. To use a nested loop structure to solve a program.

Problem A: Adjusted Guessing Game (newguessgame.py)

In the regular guessing game, a secret number in between 1 and 100 is chosen, and then the player proceeds to make guesses. After each guess, the player is told whether her guess is too high or too low. In some games with guesses, it's bad to make a guess that's too high, but okay to make a guess that is too low. In this game, the goal of the game is for the user to guess the correct number in as few guesses as possible, but if they make four guesses that are too high, they lose the game. (It's worse to lose the game than win the game in 100 guesses.)

Write a computer program that allows the user to play this game.

Input Specification

Note: It is guaranteed that whoever uses your program will adhere to these specifications. This means that you do NOT have to check for them!

Each guess will be an integer in between 1 and 100, inclusive.

Output Specification

Prompt each user for their guess with the following prompt:

```
What is your guess?
```

If the user's guess is too high, output a line with the following format:

```
Your guess is too high. You have made X guesses that were too high.
```

where X is the current number of guesses that have been made that are too high.

If the user's guess is too low, output the following line:

```
Your guess is too low.
```

If the user guesses correctly, output one final line with the following format:

```
Congrats, you got the number in G guesses, of which X were too high.
```

where G is the total number of guesses (including the winning one) the user made, and X is how many of those guesses were too high.

If the user makes four guesses that are too high and loses, output two lines with the following format:

```
Sorry you lost by making too many high guesses.
The correct number was X.
```

Output Samples

Below is one sample output of running the program. **Note that this sample is NOT a comprehensive test.** You should test your program with different data than is shown here based on the specifications given above. In the sample run below, for clarity and ease of reading, the user input is given in *italics* while the program output is in **bold**. (Note: When you actually run your program no bold or italics should appear at all. These are simply used in this description for clarity's sake.)

Sample Run #1

What is your guess?
50
Your guess is too high. You have made 1 guesses that were too high.
What is your guess?
25
Your guess is too low.
What is your guess?
30
Your guess is too low.
What is your guess?
32
Congrats, you got the number in 4 guesses, of which 1 were too high.

Sample Run #2

What is your guess?
50
Your guess is too high. You have made 1 guesses that were too high.
What is your guess?
25
Your guess is too high. You have made 2 guesses that were too high.
What is your guess?
10
Your guess is too high. You have made 3 guesses that were too high.
What is your guess?
2
Your guess is too low.
What is your guess?
5
Sorry you lost by making too many high guesses.
The correct number was 4.

Problem B: Groundhog Month Calendar (groundhog.py)

The regular calendar is quite confusing. Some months start on a Monday, others on a Wednesday, and different months have different numbers of days. (Note: of course, a month can start on any day of the week!)

The groundhogs like repeatability and would rather their calendar follow simple patterns than correspond to the Earth's rotation around the Sun.

The groundhogs label their months 1 through ***n***, where ***n*** is a positive integer. Each month has ***m*** days, where ***m*** is a positive integer, which is a multiple of 7. The groundhogs have decided that they like the seven days in a week and always want their months starting on Sunday.

Write a program that asks the user to enter in ***n*** *and* ***m*** as described above and prints out the corresponding calendar for the Groundhog year.

Input Specification

Note: It is guaranteed that whoever uses your program will adhere to these specifications. This means that you do NOT have to check for them!

n will be a positive integer in between 1 and 20.
m will be a positive integer multiple of 7 in between 7 and 70, inclusive.

Output Specification

For each month of the calendar, print a header line of output with the following format:

```
Month k
```

where k is the month number, starting with 1.

For each month of the calendar, the second line should be a single fixed string:

```
S	M	T	W	R	F	S
```

Create this string with each of the letters separated by the tab character ('\t').

Follow this with ***m***/7 rows with the day numbers, starting at 1. On each line, each number should be followed by a tab character.

At the end of each month, print out a blank line.

Output Sample

Below is one sample output of running the program. **Note that this sample is NOT a comprehensive test.** You should test your program with different data than is shown here based on the specifications given above. In the sample run below, for clarity and ease of reading, the user input is given in *italics* while the program output is in **bold**. (Note: When

you actually run your program no bold or italics should appear at all. These are simply used in this description for clarity's sake.)

Sample Run #1

```
How many months are there on the Groundhog calendar?
3
How many days per month are there on the Groundhog calendar?
28

Month 1
S     M     T     W     R     F     S
1     2     3     4     5     6     7
8     9     10    11    12    13    14
15    16    17    18    19    20    21
22    23    24    25    26    27    28

Month 2
S     M     T     W     R     F     S
1     2     3     4     5     6     7
8     9     10    11    12    13    14
15    16    17    18    19    20    21
22    23    24    25    26    27    28

Month 3
S     M     T     W     R     F     S
1     2     3     4     5     6     7
8     9     10    11    12    13    14
15    16    17    18    19    20    21
22    23    24    25    26    27    28
```

Restrictions

Please IDLE 3.6 (or higher) to develop your program. Write each in a separate file with the names specified previously, **newguessgame.py**, and **groundhog.py**

Each of your **two** programs should include a header comment with the following information: your name, course number, assignment title, and date. Also, make sure you include comments throughout your code describing the major steps in solving the problem.

Grading Details

Your programs will be graded upon the following criteria:

1) Your correctness and appropriate use of loop structures to solve the given problems. Correct solutions that are unwieldy and don't properly use loops will not earn full credit.

2) Your programming style and use of white space. Even if you have a plan and your program works perfectly, if your programming style is poor or your use of white space is poor, you could get 10% or 15% deducted from your grade.

3) Compatibility to IDLE.