CS 112 - Lab Assignment #4 Specification (If Statements and Numeric Conversions)

The purpose of this lab is to gain experience if statements, String to numeric conversions with multiple number bases. The completed lab must be submitted via Blackboard NLT (no later than) the beginning of the following week's lab session (i.e. one week after assignment). If you have questions, use the Blackboard discussion forums (and instructor/TA office hours) to obtain assistance.

Lab Requirements:

- The source code for this lab must be submitted in a file named **lab4.py**
- The source code file must contain a **file header** formatted as in previous labs.
- The source code file should use **self-documenting code** and additional comments (as required) to improve code readability.
- The solution for the lab must take input from the user on the command line.
- The solution must then present a menu with options to ask for a problem type
- The solution must display a random addition problem using Hex or binary numbers as appropriate.
- The lab must accept input from the user and determine the validity of the solution. Including a test for the right type of answer (either binary or hex)

Lab Procedure:

You are going to use the power of Python to create a program to help you learn binary and hexadecimal math.

Steps:

1. Print the following menu:

What type of problem do you want?
1. Binary
2. Hexadecimal

Type >

2. When the user answers you need to:
   Randomly choose two numbers between 0-50.
   Display an addition problem using those numbers. The display width of the number strings should be padded to 10 characters wide. Example:

```
Type >1
   0b110001
+  0b101010
------------
```

3. The user should then type in an answer in the appropriate number base. For example to answer 13 in binary the user should type: **0b1101** or for hexadecimal the user should type in: **0xD**

4. The system should check if their answer is correct. If it is correct print:
Excellent job!
(and let the program end).

If the answer is not correct print out the correct answer in the correct number base:
Wrong answer. The correct answer was: 0b111011

Hints

---

1. Generate random numbers: I would check the documentation for the random module

2. Convert a number to a hexadecimal / binary string for display:

```
>>> bin(13)
'0b1101'
>>>
>>>
>>> hex(34)
'0x22'
>>>
```

NOTE: The bin function is only available if you're in Python 2.6. If you are not you need to import a version of this function written by Prof. Fleck.

http://cs.gmu.edu/~dfleck/classes/cs112/spring09/labs/lab4/bin.py

3. Convert a user's input from hexadecimal or binary string to a number:

```
>>> eval('0x22')
34
```

>>> eval('0b1101')
13
>>>

NOTE: The eval function for binary numbers only works if you're in Python 2.6. If you are not you need to import a version of this function written by Prof. Fleck. The function is named evalBin, and is available in the following file:

http://cs.gmu.edu/~dfleck/classes/cs112/spring09/labs/lab4/bin.py

4. **To check for binary input, all you need to do is check for a "b" in the user's answer. For hexadecimal check for an "x" in the user's answer. (This is not a perfect way to check, but is fine for this lab.)**

Bonus Points

A 20% bonus will be awarded if you add in code to validate the number is a valid hex/binary number before you try to parse it, and if the number is invalid, give the user the opportunity to answer again. This loop should continue until the user enters a valid number. Meaning, you keep asking until a valid number is entered.

Be sure to note any bonus options you created in the file header of your program.

Sample Program Output: (Bolded is user-typed input)

...spring09/non_web/lab_solutions/lab4 > python lab4.py

```
What type of problem do you want?
1. Binary
2. Hexadecimal


Type >1
    0b10110
+     0b101
------------
0b11011
Excellent job!
```

...spring09/non_web/lab_solutions/lab4 > python lab4.py

```
What type of problem do you want?
1. Binary
```

```
2. Hexadecimal


Type >1
     0b1001
+  0b101010
------------
0b1101
Wrong answer. The correct answer was: 0b110011


...spring09/non_web/lab_solutions/lab4 >  python lab4.py
What type of problem do you want?
1. Binary
2. Hexadecimal


Type >2
       0x1a
+      0x1e
------------
0x1f
Wrong answer. The correct answer was: 0x38

...spring09/non_web/lab_solutions/lab4 >  python lab4.py

What type of problem do you want?
1. Binary
2. Hexadecimal


Type >2
       0x2a
+      0x30
------------
0x5a
Excellent job!


>>>

What type of problem do you want?
1. Binary
2. Hexadecimal


Type >2
       0x16
+       0xb
------------
```

```
19
You must answer in hex!

>>>
What type of problem do you want?
1. Binary
2. Hexadecimal


Type >1
    0b11110
+   0b10100
------------
13
You must answer in binary!
```

Lab Assignment #4

	Excellent (85% or higher)	Average (60% or higher)	Needs Improving (Less then 60%)	Points Possible
Core Concepts (Topics of Focus)	• A menu is shown to read input from the user. Including correctly aligning the numbers. • Printed output to user. • Code accurately converts user input to a number • Code accurately converts output to the correct numeric string • Structure the program is clean.	• Some number conversions are missing (decimal numbers are displayed) • Numeric calculations are incorrect (additions are done in decimal, thereby not being correct for the output.) • Used functions, but in a poorly organized way.	• No number conversions present • Additions not implemented • Failed to read input from user. • Failed to print output to user.	5
User Interface / Input/Output	• Code correctly prompts the user for the appropriate inputs. • Menu is displayed and accepts only the correct inputs and calls an appropriate function. • Output is descriptive to the user and is well formatted. • Input from the users is checked for a valid type (bin or hex) as the answer for the problem.	• Input prompt does not tell the user what to enter • Output does not match the specification • Output is not checked for valid type (bin or hex)	• No menu is present or options do not work. • Program does not prompt for input	3
Syntax / Overall Coding Guidelines	• Code runs without any errors. • Comments are meaningful and professional. • Code is clean and easy to read. Uses meaningful variable names • Complex algorithms (three lines or more) are commented with information on purpose and start/stop states.	• Comments are not helpful for understanding the code • variable names are not meaningful • code generates errors in edge cases	• Code generates errors in common cases • no comments were used	2
Bonus Points	Code correctly rejects any invalid binary/hex number and reprompts the user for a correct number..	None given.	None given.	(+2)
Final Score				10 (+2)

Additional Comments: