CSci 1133, Spring 2015
Lab Exercise 3

Selective Control

This week, we begin exploring the dynamic facilities that support the *conditional* execution of selected program statements. Imperative programming languages such as Python provide high-level constructs that support *conditional selection* and *iteration*. *Conditional selection* refers to the computational facility that *conditionally* executes statements or groups of statements based on the outcome of a dynamic test. *Conditional iteration* refers to the ability to repeatedly execute statements or groups of statements while some dynamically tested criteria remains satisfied.

Warm-up

1) **Basal Metabolic Rate** (adapted from Walter Savitch, Problem Solving with C++)

The Harris-Benedict equation estimates the number of calories your body needs to maintain your weight if you do no exercise. This is called your *basal metabolic rate*, or BMR. The formula for the calories needed for a woman to maintain her weight is:

BMR = 655 + (4.3 * weight in pounds) + (4.7 * height in inches) – (4.7 * age in years)

The formula for the calories needed for a man to maintain his weight is:

BMR = 66 + (6.3 * weight in pounds) + (12.9 * height in inches) – (6.8 * age in years)

A typical chocolate bar will contain around 230 calories. Write a program that allows the user to input his or her weight in pounds, height in inches, age in years, and the character 'M' for male or 'F' for female. The program should then output the number of 230 calorie chocolate bars that need to be consumed to maintain one's weight for the appropriate sex of the specified height, weight and age.

2). **Rounding floating-point values**

In lecture, we discussed *explicit* type conversion in Python using the functions: `str()`, `float()` and `int()`. Note that if you convert a floating-point value to an integer using the `int()` function, the result will be truncated, not *rounded* as one might expect. i.e.,

```
>>> x = 4.67
>>> xi = int(x)
>>> print(xi)
 4
```

One way to round a floating-point number to the *nearest* integer (data type `int`) is to first add 0.5 if the number is positive or subtract 0.5 if the number is negative, then truncate (discard the fractional part of) the result.

a). Write a Python program that will input a floating-point value, then round it to the nearest integer and output the result.

b). Note that as your programs become increasingly more complex, it is essential to verify they work as intended. You need to carefully consider what set of input values should be tried in order to convince yourself that your program produces the correct answer in *all* cases. One important consideration is *path coverage*. A single branch *path* consists of the instructions that are executed when one "path" of an if-else statement is taken. This is dependent on the value of the conditional test. Path coverage involves providing inputs that ensure sure *every* branch path has been executed.

How many runs (test cases) should you make, at a minimum, to verify that this program is running correctly? What data should you input for each of the test cases and what is the expected result? **Discuss your answer with one of your TAs and demonstrate the result of your program for each test case.**

Stretch

1). **Metric Conversions**

Write a Python program that will Convert metric measurements to/from Imperial units as follows:

pounds to/from kilos
ounces to/from grams
miles to/from kilometers
inches to/from centimeters

Your program should do the following: input two values: a floating point measurement and a separate "units" string from the following set:

{'pounds', 'ounces', 'miles', 'inches', 'kilos', 'grams', 'kilometers', 'centimeters'}

then convert the value to the corresponding units and output both the converted value and its units.

If the units string is not recognized (not in those specified), your program should produce an error message

Example:

```
>>> Enter value: 1.0
>>> Enter units: centimeters
>>> 0.3937 inches

>>> Enter value: 10
>>> Enter units: ounces
>>> 283.5 grams

>>> Enter value:
>>> Enter units: 100 grms
>>> 100 grms?
```

2). **Largest of Three Values**

Write a Python program that will input three floating-point values (x, y, z) then determine and output the largest of the three values. Identify and test all possible cases to ensure your program works correctly.

Workout

1). **Day of the Week**

Zeller's congruence is a method to determine the day of the week (e.g., Sunday, Monday, etc.) from a calendar date given the day (*d*), *modified-month* (*m*) and year (*y*):

$$\left[d + 5 + \frac{26(m+1)}{10} + \frac{5(y\,\%\,100)}{4} + \frac{21(y/100)}{4}\right] \%\,7$$

Note that all division is *truncated* division. The *modified-month* value only applies to the months of January and February, which are counted as months 13 and 14 of the *previous* year: e.g., 13 = Jan, 14 = Feb, 3 = Mar, 4 = Apr, 5 = May, . . . 12 = Dec. For example, 2/21/2014 (Feb 21, 2014) would be represented as 14/21/2013.

Zeller's congruence gives the day of the week starting with 0 = Monday, 1 = Tuesday and so on.

Write a program that will input a date as three separate integer values (month, day and year) and return the day-of-week value as determined by *Zeller's* method.

Example:

```
Enter month: 2
Enter day: 28
Enter year: 2014
2/28/2014 is a Friday

Enter month: 2
Enter day: 28
Enter year: 2000
2/28/2000 is a Monday

Enter month: 2
Enter day: 29
Enter year: 2000
2/29/2000 is a Tuesday
```

Challenge

Try these challenge problems if you have extra time or would like additional practice outside of lab.

1). **What is Tomorrow's Date?**

Write a Python program that will ask the user to input *month*, *day* and *year* values (as 3 separate integers), and output the month, day and year values for the *following* day. (Hint: you will need to determine the number of days in the month, which (for one month) is dependent on whether the year is a leap year or not. You will also have to consider how to handle December 31)

2). **XOR**

The "exclusive-or" logical operation (XOR) is an operation on two Boolean operands A, B that produces true iff (if-and-only-if) A is *different* than B. Python does not have an exclusive-or (XOR) logical operation, however you can easily construct XOR using `and`, `or` and `not` operations.

Truth Table for XOR:

A	B	A XOR B
F	F	F
F	T	T
T	F	T
T	T	F

In usage, exclusive-or indicates: "A or B, but not both". This is consistent with our typical use of spoken language. For example, you might say "I'm going to the movies or bowling on Friday evening". What you actually mean is that you will either be going bowling or to the movies, but not both (exclusive-or).

a). Construct an XOR expression using the Python operators `and`, `or` and `not`:

b). Devise and write a short Python program to test your logical expression.