CSC1015F Assignment 4: Functions (and strings)

Assignment Instructions

This assignment involves constructing Python modules that use input and output statements, '`if`' and '`if-else`' control flow statements, '`while`' statements, '`for`' statements, statements that perform numerical and string manipulation, and functions.

Question 1 [30 marks]

Write a Python module called '`calutils.py`', and we need you to write this.

The module should contain the following functions:

- `is_leap_year(year)`
  Given a year (a 4 digit number), returns true if it is a leap year, and false otherwise.
- `month_name(number)`
  Given the number of a month, returns the corresponding name. 1 is January, ..., 12 is December.
- `days_in_month(month_number, year)`
  Given a month (in the form of a number) and (4 digit) year, return the number of days in the month (accounting, in the case of February, for whether or not it is a leap year).
- `first_day_of_year(year)`
  Given a 4 digit year, return the number of the day on which January 1st falls. 0 is Sunday, …, 6 is Saturday. (See question 2 of assignment 2.)
- `first_day_of_month(month_number, year)`
  Given a month (in the form of a number) and (4 digit) year, return the number of the day on which the first of that month falls. 0 is Sunday, …, 6 is Saturday.

  In each case we've given the name of the required function and its parameters.

On the Vula assignment page you will find a program called '`calutilsmodtest.py`'. You can use this to check that your module functions correctly. Here is a sample transcript:

```
Choose from the following options:
0. quit
1. Test is_leap_year().
2. Test month_name().
3. Test days_in_month().
4. Test first_day_of_year().
5. Test first_day_of_month().
1
Enter the year (4 digits):
2016
The year 2016 is a leap year.

Choose from the following options:
0. quit
1. Test is_leap_year().
2. Test month_name().
3. Test days_in_month().
4. Test first_day_of_year().
5. Test first_day_of_month().
2
```

CONTINUED

```
Month number 1 is January.
Month number 2 is February.
Month number 3 is March.
Month number 4 is April.
Month number 5 is May.
Month number 6 is June.
Month number 7 is July.
Month number 8 is August.
Month number 9 is September.
Month number 10 is October.
Month number 11 is November.
Month number 12 is December.

Choose from the following options:
0. quit
1. Test is_leap_year().
2. Test month_name().
3. Test days_in_month().
4. Test first_day_of_year().
5. Test first_day_of_month().
0
```

Question 2 [35 marks]

Mathematical functions map naturally to program functions and modules often are used to group such functions for reuse.

In the Bukiyip* language, coconuts, days and fish are counted in base 3. Numbers use only the digits 0-2, such that instead of "tens" and "hundreds", the second and third digits represents multiples of 3 and 9 respectively.

(Reference: http://mentalfloss.com/article/31879/12-mind-blowing-number-systems-other-languages)

Write a Python module called '`bukiyip.py`' with the following functions for simple Bukiyip arithmetic, assuming that all values have at most 3 digits.

- `bukiyip_to_decimal(a)`
  Convert a Bukiyip number to decimal.
- `decimal_to_bukiyip(a)`
  Convert a decimal number to Bukiyip.
- `bukiyip_add(a, b)`
  Add two Bukiyip numbers.
- `bukiyip_multiply(a, b)`
  Multiply two Bukiyip numbers.

A main program has been supplied as '`bukiyipmodtest.py`' - use this to test your program and do not change this file.

Sample I/O:

```
**** Bukiyip test program ****
Available commands:
d <number> : convert given decimal number to base-3.
b <number> : convert given base-3 number to decimal.
```

CONTINUED

```
a <number> <number> : add the given base-3 numbers.
m <number> <number> : multiply the given base-3 numbers.
q : quit

Enter a command:
d 12
110
Enter a command:
b 20
6
Enter a command:
a 12 11
100
Enter a command:
m 12 11
202
Enter a command:
q
```

Question 3 [35 marks]

Write a Python module called `piglatin.py` that contains functions for translating sentences between English and a variant of Pig Latin (see: http://en.wikipedia.org/wiki/Pig_Latin).

To convert from English to Pig Latin, each word must be transformed as follows:

- if the word begins with a vowel, 'way' should be appended (example: 'apple' becomes 'appleway')
- if the word begins with a sequence of consonants, this sequence should be moved to the end, prefixed with 'a' and followed by 'ay' (example: 'please' becomes 'easeplay')

NB: Assume, when reverting Pig Latin to English that the original English text does not contain the letter "w".

The Python module will contain the following functions:

- `to_pig_latin(sentence)`
  Return the Pig Latin sentence for a given English sentence.
- `to_english(sentence)`
  Return the English sentence for a given Pig Latin sentence.

A main program called '`plmodtest.py`' has been provided. Use this to test your program. (Note `plmodtest` must not be modified.)

Sample I/O:

```
(E)nglish or (P)ig Latin?
E
Enter an English sentence:
the quick black fox jumps over the lazy apple
Pig-Latin:
eathay uickaqay ackablay oxafay umpsajay overway eathay azyalay
appleway
```

Sample I/O:

```
(E)nglish or (P)ig Latin?
P
Enter a Pig Latin sentence:
eathay uickaqay ackablay oxafay umpsajay overway eathay azyalay
appleway
English:
the quick black fox jumps over the lazy apple
```

Submission

Create and submit a Zip file called '`ABCXYZ123.zip`' (where ABCXYZ123 is YOUR student number) containing `calutils.py`, `bukiyp.py`, and `piglatin.py`.

END

CONTINUED