6.189 Homework 1, Optional Problems

http://web.mit.edu/6.189/www/materials.html

Exercise OPT.1 – Zeller's Algorithm

OPTIONAL!- Some problem sets will have optional exercises at the end. Feel free to work on these problems if you have time at the end of the assignment, but you certainly don't have to do them. However, you will get excellent practice in Python, and we will give you feedback on any optional work you turn in.

Zeller's algorithm computes the day of the week on which a given date will fall (or fell). In this exercise, you will write a program to run Zeller's algorithm on a specific date. You will need to create a new file for this program, `zellers.py`. The program should use the algorithm outlined below to compute the day of the week on which the user's birthday fell in the year you were born and print the result to the screen.

Start with the program in Exercise 1.4, but ask for the month as a number between 1-12 where March is 1 and February is 12. If born in Jan or Feb, enter previous year (see the notes below). In the end, print out the name of the user and on what they of the week they were born.

Zeller's algorithm is defined as follows:

Let A, B, C, D denote integer variables that have the following values:

```
A = the month of the year, with March having the value 1, April the
value 2, . . ., December the value 10, and January and February being
counted as months 11 and 12 of the preceding year (in which
case,subtract 1 from C)
B = the day of the month (1, 2, 3, . . . , 30, 31)
C = the year of the century (e.g. C = 89 for the year 1989)
D = the century (e.g. D = 19 for the year 1989)
```

Note: if the month is January or February, then the preceding year is used for computation. This is because there was a period in history when March 1st, not January 1st, was the beginning of the year.

Let W, X, Y, Z, R also denote integer variables. Compute their values in the following order using integer arithmetic:

```
W = (13*A - 1) / 5
X = C / 4
Y = D / 4
Z = W + X + Y + B + C - 2*D
R = the remainder when Z is divided by 7
```

The value of R is the day of the week, where 0 represents Sunday, 1 is Monday, . . ., 6 is Saturday. If the computed value of R is a negative number, add 7 to get a non negative number between 0 and 6 (you

don't need to do this in the code). Print out R. You can check to be sure your code is working by looking at `http://www.timeanddate.com/calendar/`.

Run some test cases- try today's date, your birth date, and whatever else interests you!

Feel free to show your `zellers.py` code to a staff member (although if it's close to a checkoff due date, or the lab is really busy, we may ask you to come back later), and we'll look it over with you!

Exercise OPT.2 – Secret Messages

OPTIONAL! This exercise is tricky! Be sure to ask the LAs for help if you need them!

The goal of this exercise is to write a cyclic cipher to encrypt messages. This type of cipher was used by Julius Caesar to communicate with his generals. It is very simple to generate but it can actually be easily broken and does not provide the security one would hope for.

The key idea behind the Caesar cipher is to replace each letter by a letter some fixed number of positions down the alphabet. For example, if we want to create a cipher shifting by 3, you will get the following mapping:

```
Plain:   ABCDEFGHIJKLMNOPQRSTUVWXYZ
Cipher:  DEFGHIJKLMNOPQRSTUVWXYZABC
```

To be able to generate the cipher above, we need to understand a little bit about how text is represented inside the computer. Each character has a numerical value and one of the standard encodings is ASCII (American Standard Code for Information Interchange). It is a mapping between the numerical value and the character graphic. For example, the ASCII value of 'A' is 65 and the ASCII value of 'a' is 97. To convert between the ASCII code and the character value in Python, you can use the following code:

```
letter = 'a'

# converts a letter to ascii code
ascii_code = ord(letter)

# converts ascii code to a letter
letter_res = chr(ascii_code)

print ascii_code, letter_res
```

Start small. Do not try to implement the entire program at once. Break the program into parts as follows:

1. Create a file called `cipher.py`. Start your program by asking the user for a phrase to encode and the shift value. Then begin the structure of your program by entering in this loop (we'll build on it more in a bit):

   ```
   encoded_phrase = ''

   for c in phrase:
      encoded_phrase = encoded_phrase + c
   ```

   What does this loop do? Make sure you understand what the code does *before* moving on!

2. Now modify the program above to replace all the alphabetic characters with '`x`'. For example:

```
Enter sentence to encrypt: Mayday! Mayday!
Enter shift value: 4
The encoded phrase is:  Xxxxxx! Xxxxxx!
```

We are going to apply the cipher only to the alphabetic characters and we will ignore the others.

3. Now modify your code, so that it produces the encoded string using the cyclic cipher with the shift value entered by the user. Let's see how one might do a cyclic shift. Let's say we have the sequence:

   ```
   012345
   ```

   If we use a shift value of 4 and just shift all the numbers, the result will be:

   ```
   456789
   ```

   We want the values of the numbers to remain between 0 and 5. To do this we will use the modulus operator. The expression x%y will return a number in the range 0 to y-1 inclusive, e.g. 4%6 = 4, 6%6 = 0, 7%6 =1. Thus the result of the operation will be:

   ```
   450123
   ```

   Hint: Note that the ASCII value of '`A`' is 65 and '`a`' is 97, not 0. So you will have to think how to use the modulus operator to achieve the desired result. Apply the cipher separately to the upper and lower case letters.

   Here is what you program should output:

   ```
   Enter sentence to encrypt: Mayday! Mayday!
   Enter shift value: 4
   The encoded phrase is:  Qechec! Qechec!
   ```

   Feel free to show your `cipher.py` code to a staff member (although if it's close to a checkoff due date, or the lab is really busy, we may ask you to come back later), and we'll look it over with you!