COSC 101 Homework 6 Fall 2014

Due date: **Thursday, October 16, 11:55pm**

Program 1: Leap Year

Your task is to write a program `hw6_leap.py` that asks the user for a range of years and then prints for each year within that range if it is a leap or a normal year. Since 1582, a leap year occurs according to the following formula: a leap year is divisible by four, but not by one hundred, unless it is divisible by four hundred.

First write a function `is_leap_year(y)` that returns `True` if the the parameter `y` is a leap year and `False` if `y` is a normal year. Test this function on its own.

Now write the main program that asks the user for a year range and first checks that the range starts at or after 1582. If the range is valid, your program output the type for each year in ascending order using the `is_leap_year` function you wrote. Important detail: you must handle the case when the user enters the years "out of order" in the sense that the first input is the end of the range (the later year) and and the second input is the start of the range (the earlier year).

Your program should reproduce the following three executions. In the first exexcution the user enters `2008` followed by `2024`.

```
Enter a year: 2008
Enter a second year: 2024
2008 is a leap year
2009 is a normal year
2010 is a normal year
2011 is a normal year
2012 is a leap year
2013 is a normal year
2014 is a normal year
2015 is a normal year
2016 is a leap year
2017 is a normal year
2018 is a normal year
2019 is a normal year
2020 is a leap year
2021 is a normal year
2022 is a normal year
2023 is a normal year
2024 is a leap year
```

In this example, the user enters the years "out of order" but the program still correctly prints the range.

```
Enter a year 2400
Enter a second year 2392

2392 is a leap year
2393 is a normal year
2394 is a normal year
2395 is a normal year
2396 is a leap year
```

```
2397 is a normal year
2398 is a normal year
2399 is a normal year
2400 is a leap year
```

Lastly, the user enters an invalid range.

```
Enter a year 1000
Enter a second year 1004
The range must start after or at 1582
```

Program 2: Farm song

Write a program `hw6_farm.py` that prints the lyrics of the song Old MacDonald. The program first asks the user the name of the farmer and the number of the various animals the farmer keeps on the farm. Then the user is prompted to enter the name for each animal type and the noise it makes. The program prints the lyrics of the song accordingly.

You program should use at least two functions. One function that collects the animal types and noises and one function that builds one verse of the song. **Lists should not be used in your solution.**

An example of the program execution is as follows:

```
What is the farmer name?    Maturin
How many different animals does Maturin keeps on the farm?  2

Enter the name of animal 1:  cow
Enter the noise of animal 1:  moo
Enter the name of animal 2:  duck
Enter the noise of animal 2: quack

Maturin had a farm, E-I-E-I-O.
And on that farm he had a cow, E-I-E-I-O.
With a moo moo here and a moo moo there
Here a moo, there a moo, everywhere a moo moo
Maturin had a farm, E-I-E-I-O.

Maturin had a farm, E-I-E-I-O.
And on that farm he had a duck, E-I-E-I-O.
With a quack quack here and a quack quack there
Here a quack, there a quack, everywhere a quack quack
Maturin had a farm, E-I-E-I-O.
```

In the above example, the user entered six inputs `Maturin`, `2`, `cow`, `moo`, `duck`, and `quack`. Your program must match the format above *exactly*.

Program 3: Anagram

Your task is to write a program `hw6_anagram.py` that checks if two strings are anagrams. An anagram is a type of word play, the result of rearranging the letters of a word or phrase to produce a new word or phrase, using all the original letters exactly once.

First write a `remove_single` method that takes a string and a character and returns the parameter string with the character removed if it exists. If the character appears multiple times in the string, only a single occurrence is removed.

For example

```
print remove_single('anagram', 'a')
> nagram
print remove_single(remove_single(remove_single('anagram', 'a'), 'a'), 'a')
> ngrm
```

Next write a function `is_anagram` that takes two strings and uses `remove_single` to determine if the two strings are anagrams: `is_anagram` returns `True` if the two strings have exactly the same letters and may only differ by white spaces and returns `False` if there are any difference in letters.

For example

```
print is_anagram("orchestra", "carthorse")
> True
print is_anagram("orchestra", "courthouse")
> False
print is_anagram("a decimal point", "im a dot in place")
> True
print is_anagram("debit card", "bad credit")
> True
print is_anagram("snooze alarms",   "alas no more zzs")
> False
```

Finally, write a main program that asks the user for two inputs and tells the user whether or not they are anagrams. Here are two example executions.

```
Enter the first phrase: dormitory
Enter the second phrase: dirty room
Yes, these are anagrams!

Enter the first phrase: dormitory
Enter the second phrase: clean room
Sorry, these are not anagrams!
```