1. Write a function called date_to_string that takes three integer arguments, called month, day, and year. This function returns a nicely-formatted date as a string, with an abbreviated month, and appropriate punctuation. You may need to use the function str(n), which takes an integer argument n and returns it as a string. For instance str(23) returns "23".

   Ex: date_to_string(10, 27, 2015) returns "Oct 27, 2015"

2. Write a program that lets the user enter months, days, and years as integers from the keyboard, and prints the corresponding dates as strings using your date_to_string function. Let the user keep entering new dates until they enter Halloween of any year. Use input validation so they never enter an invalid date (check that the day of the month they enter is appropriate for the month; that is, if they choose a month with 30 days, they shouldn't be able to enter 31 as the day. Checking for Feb 29 only on leap years is optional).

3. Write a function called date_earlier that takes six integer arguments, corresponding to the month, day, and year of two different dates. This function returns True if the first date is earlier than the second.

   Ex: date_earlier(10, 27, 2013, 12, 12, 2013) returns True.
   Ex: date_earlier(10, 27, 2013, 1, 30, 2013) returns False.

4. Write a program that opens the file dates.txt [download from see class webpage] and prints each date inside as a string. The program should also print the earliest date in the file and the latest date in the file.

5. Write a function called count_divisors that takes a single integer parameter called n. This function should return the number of divisors that the number n has. A divisor of n is a number x such that n/x is an integer. Hint: use the % operator to test divisibility, and use a loop to count the divisors.

   Ex: count_divisors(10) returns 4 because the divisors of 10 are 1, 2, 5, and 10.

6. Write a function called is_prime that takes a single integer parameter called n. This function returns True if a number is prime. Do not call count_divisors for this. Instead, create a loop that runs from 2 to the square root of n and tests each number in that range to see if it's a divisor. As soon as a divisor is found, end the loop and return False. If no divisors are found, then the number is prime, and you would return True. (Work this logic out on paper with some small values of n to see how this algorithm works before you write it.)

7. Write a function called is_on_campus that accepts a string parameter called phone. The parameter is a phone number with either 10 digits or 7 digits. Any 10 digit phone number will look like "aaa-ddd-dddd" where d is a digit and aaa is a 3-digit area code. Any 7 digit phone number will look like "ddd-dddd". This function should examine the phone number to see if it corresponds to a Rhodes College phone (all college phone numbers' first three digits are 843). Return True if the phone number is on-campus, and False otherwise.

   Ex: is_on_campus("901-843-3725") returns True.
   Ex: is_on_campus("703-843-2487") returns False (wrong area code).
   Ex: is_on_campus("843-3587") returns True (assume 901 area code if none is given).
   Ex: is_on_campus("238-8431") returns False (doesn't start with 843).

8. Write a program that lets the user type in a string from the keyboard. Use a loop to count the number of letters in the string, and the number of numbers. So if the user types in the string "abc123xyz4", your program should report that the string has 6 letters and 4 numbers.

   Hint: If s is a string, then s.isalpha() returns True if s is a letter, and s.isnumeric() returns True if s is a digit.