22C:16 Homework 2

Due via ICON on Wednesday, Feb 9th, 4:59 pm

1. Here is the second Python program that we have examined in detail in the lectures.

```
import math
n = int(raw_input("Enter a positive integer:"))

isPrime = True
factor = 2
factorBound = math.sqrt(n)

while factor <= factorBound:
    if n % factor == 0:
        isPrime = False
        break
    factor = factor + 1

if isPrime:
    print n, "is a prime"
else:
    print n, "is a composite"
```

   (a) List all the variables that this program uses. Also mention the type of each variable. A variable may change its type during the course of execution and if that is happening for any variable in this program, make sure that you mention this explicitly.

   (b) Suppose that this program is executed and when prompted for a positive integer, the user types `65`. Write down all the values that the variable `factor` takes on (in order) over the course of execution of the program.

2. Modify the above program (from Problem 1) so that if `n` is found to be a composite, the program also outputs a factor of `n`, as evidence of the fact that `n` is composite. For example, if `n` were `25`, the output message might be

   ```
   25 is a composite; I found factor 5
   ```

3. Explain in 2-3 sentences what the role the `break` statement in the above program is. Your answer should address the following issues: (i) how does the `break` statement affect the flow of control of the program, (ii) would the program work correctly if we simply deleted the `break` statement, and (iii) how does the `break` statement improve the program.

4. Suppose that `x` is a variable with value 10. Evaluate the values of the following boolean expressions.

   (a) `(x < 20) and (x < 0)`

   (b) `(x < 20) or (x < 0)`

   (c) `(x < 0) or (x > 50)`

(d) `not (x < 0)`

(e) `not (x != 0)`

(f) `(x < 20) and (x != 50)`

5. Consider the following program.

```
n = int(raw_input("Enter a positive integer:"))
m = int(raw_input("Enter a positive integer:"))

while n <= m:
    if n % 7 == 0:
        break
    print n, "is the current value of the variable n"
    n = n + 1
```

   (a) What is the output you will see if you run this program and provide input (in this order) 37 and 100.

   (b) Make a program without a `break` statement that is equivalent to the above program. In other words, the new prgram should behave exactly like the above program on all inputs. You should do this by using a new boolean variable that "remembers" when `n` has become a multiple of 7. This boolean variable will help your program exit the loop immediately.

6. Write a program that prompts the user for a sequence of positive integers and then outputs the number of integers in the sequence. The user will input `0` to indicate that she is finished inputting her sequence of positive integers. The `0` is not considered part of the sequence that your program needs to process Here is an example interaction between the user and the program.

```
Enter a number: 7
Enter a number: 9
Enter a number: 90
Enter a number: 3
Enter a number: 0
Length of sequence: 4
```

7. Write a program that prompts the user for a sequence of strings and then outputs the longest string input by the user. The user will simply hit `return` without entering any string to indicate that she is done. There may be several longest strings; in this case your program can break ties arbitrarily. Here is an example interaction between the user and the program.

```
Enter a string: hi
Enter a string: hello
Enter a string: bye
Enter a string: heard
Enter a string: test
Enter a string:
The longest string is hello
```