22C:16 Exam 1

Feb 22nd, 6:30 pm to 8:30 pm

Instructions:

- This is an open notes exam and you have 2 hours to complete it. There are 4 problems in the exam and these appear on 7 pages.
- Make sure that you do not have any electronic devices (phones, calculators, etc.) on your desk; you are not allowed to access these during your exam.
- Write as neatly as you can.
- Show all your work, if you want to receive partial credit.

Name:

Circle your section:

SCA (M, W evening)	A01 (9:30-10:20)	A02 (11-11:50)	A03 (12:30-1:20)
	A04 (12:30-1:20)	A05 (2-2:50)	A06 (3:30-4:20)

1. [**30 points**] For each of the following expressions, first determine if evaluating it would cause Python to report an error; if so, explain in a sentence what it was about the expression that caused the error. Otherwise, if you think the expression will evaluate in Python in an error-free manner, write down the *value* that the expression evaluates to and its *type*. Assume that the modules `math`, `sys`, and `random` have already been imported prior to the evaluation of these expressions.

   (a) `bin(len(str(10<20)))`

   (b) `4L*len("Problem1")/(len("Exam1")%3)+2.0`

   (c) `len('twelve'+"four")/(len('2L')+2*(-1)**3)`

(d) `round(math.pi, 2) < math.trunc(math.pi)`

(e) `20 < 30 and 50 < 25 and 4 < 100/0`

(f) `str(12 + 4.0/2**2)`

(g) `3.0 * input("Enter an integer:  ")`
(Assume that user inputs the expression `10/2 + 1`, when prompted)

(h) `not bool(0.1) or not (100 and not False)`

(i) `2*float("sys.maxint")`

(j) `len(str(random.randint(1, 4)*22))`

2. [**40 points**] For each of the programs below, write down the output produced by the program.

(a) What output does this program produce?

```
x = 10
while x < 120:
    y = x + 40
    while (x < y) :
        if (y % 10) == 5 :
            y = y + 15
        else :
            print x, y
            y = y - 35
    x = x + 30
```

(b) What output does this program produce?

```
x = 0
while x < 10:
    if x % 2 == 0:
        y = x + 1
        while (y < 11) :
            print "Line 1:", x, y
            y = y + 4
    else :
        y = 11-x
        while (y > 1) :
            print "Line 2:", x, y
            y = y - 3
    x = x + 3
```

(c) For this problem, assume that the user types 2079 when prompted to type something. What output does this program produce?

```
n = int(raw_input("Please type something."))

answer = ""
while n > 0:
        if n % 10 > 0:
                answer = str(n) + answer
        print answer
        n = n/10
```

(d) For this problem, assume that the user types 8033, 3443, 2117, 2332, and 0, in different lines. What output does this program produce?

```
print "Type positive integers, one per line and then 0 to be done."
n = 1
while n:
    n = int(raw_input())
    if (n/1000 == n%10) and ((n/100)%10 == (n%100)/10):
        print n
```

3. [**40 points**] In this problem, you are given two partially completed programs. Your task is to complete each program.

(a) In class we designed a program that translated numbers in base-10 (i.e., decimal representation) into equivalent numbers in base-2 (i.e., binary representation). In this problem, your task is to complete a program that translates numbers in binary into equivalent decimal numbers. Here is an example of how your program should interact with the user:

```
Please type an integer in binary: 101001
The decimal equivalent of 101001 is 41
```

In this interaction, the user types `101001`. The rest of what you see was produced by the program.

Recall that the decimal equivalent of the binary number $b_{n-1} \cdots b_2 b_1 b_0$ is

$$b_{n-1} \cdot 2^{n-1} + \cdots + b_1 \cdot 2^1 + b_0 \cdot 2^0.$$

For example, the decimal equivalent of 101001 is

$$1 \cdot 2^5 + 0 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 32 + 8 + 1 = 41.$$

Here is a partially completed Python program that uses the above-mentioned formula to compute the decimal equivalent of a binary number. Complete the following program by filling in the two blanks in it.

```
b = int(raw_input("Type an integer in binary. "))
bCopy = b
answer = 0
placeValue = 1 # tracks place values, from right to left

while b:
    bit = b % 10 # extract the right most bit

    # update the answer using the right most bit (Blank 1)

    ----------------------------------------------------

    # update b (Blank 2)

    ----------------------------------------------------

    # update placeValue
    placeValue = 2 * placeValue

print "The decimal equivalent of", bCopy, "is", answer
```

(b) Here is a partially completed program that aims to solve the following problem. The user types in a sequence of positive integers, one per line, ending with the number `0`. The program reads in this sequence and counts and outputs the number of *pairs of consecutive numbers that are in increasing order*. An example interaction of this program with the user is given below.

```
Type a positive int (zero if done).20
Type a positive int (zero if done).23
Type a positive int (zero if done).25
Type a positive int (zero if done).20
Type a positive int (zero if done).19
Type a positive int (zero if done).9
Type a positive int (zero if done).10
Type a positive int (zero if done).0
3
```

The program outputs 3 because it detects three pairs of consecutive numbers in increasing order: (i) $20, 23$, (ii) $23, 25$, and (iii) $9, 10$. The program below has two blanks to fill. Your task is to fill in these blanks.

```
import sys

# Variable used to read input numbers
current = int(raw_input("Type a positive int (zero if done)."))

# tracks the number just prior to the most recently read number
previous = sys.maxint

# counter to track the number of consecutive, increasing pairs
numIncreasingPairs = 0

while current:
    if current > previous:
        numIncreasingPairs = numIncreasingPairs + 1

    # Update previous (Blank 1)

    ---------------------------------------------

    # Update current (Blank 2)

    ---------------------------------------------------------

print numIncreasingPairs
```

4. [**40 points**] I have defined a function called `startsWith` whose header is given below:

```
def startsWith(word, letter):
```

Both parameters `word` and `letter` are expected to be of string type and furthermore the second parameter, i.e., `letter` is expected to be of length 1. The function returns `True` if the first character in `word` happens to be equal to `letter`; otherwise the function returns `False`. For example, the function call `startsWith("hello", "x")` returns `False` and the function call `startsWith("fellow", "f")` returns `True`.

(a) Define a function `startsWithDigit` that takes a single parameter `word` and returns `True` if the given `word` starts with "0", "1", "2", ..., "9". The function `startsWithDigit` should repeatedly and appropriately call the given function `startsWith` to complete its task.

(b) Write a program that prompts a user for their "password" and then determines if the password is *valid* or not. A password is said to be *valid* if it starts with a digit **and** it has length 6 or more. If your program determines that the user-entered password is not valid, it should print a message saying so. Otherwise, it should print a message saying that it has accepted the user-entered password. Your program should make use of the function `startsWithDigit` defined above.