Introduction to Computer Programming — **Problem Set #0**

Instructions

- Do the problem set entirely on your own. Questions should be directed to the instructor and no one else.
- Write your name on the top of the first page (this page).
- When asked, provide clear, *concise* explanations.
- *Handwrite* solutions in **black ink**. If you have trouble writing legibly (I sympathize!) then you may *staple* one or more typed pages (printed in *black ink* on *otherwise blank paper*) to your problem set. Some problems require you to annotate the problem itself (e.g., to circle parts of some code).

---

1 expressions and statements

Consider the following code fragment:

```
1   n = int(input('enter a number: '))
2   t = 0
3   i = 1
4   while i <= n:
5       if i % 2 == 1:
6           t += i
7       else:
8           t += i
9       i += 1
10  print(t)
```

a. Circle all of the Boolean-valued expressions.

b. Circle the line numbers that correspond to the first line of each *compound* statement.

c. In a clear and concise sentence, what value does this code print on its final line (in terms of `n`)?

d. Would you describe this as an efficient way to achieve that computation? Explain.

2 binary numbers

a. Express the binary number **11101** in base 10. Show your work.

b. Express **38** in binary. Show your work.

c. Express the binary number **10101110** in base 10. Show your work.

d. Express **241** in binary. Show your work.

3 errors

a. For each code fragment, indicate (briefly!) why it might result in a run-time error.

```
a % b
```

```
c = int(input('enter a number: '))
```

```
d = input('enter some text: ')
e = d[1]
```

```
f = open('words.txt')
```

---

b. Consider the following function:

```
def find_first_z(s):
    """Returns the index in s at which the letter z (lowercase or uppercase)
    first occurs. Returns None if no z exists in the string.
    >>> find_first_z('Bronx Zoo')
    6
    >>> find_first_z('none here!') is None
    True
    """
    k = 0
    z_pos = None
    while z_pos is None or k < len(s):
        if s[k] == 'z' or s[k] = 'Z':
            z_pos = k
            k += 1
    return pos_z
```

Identify errors by circling (or highlighting) specific parts of code. For each, briefly explain its effect and how to fix it.

4 evaluating expressions

Assume the following lines have just been executed in order (and that `random` has been imported):

```
a = 22
b = 'true'
c = input('Shall we play a game?')
d = 'side'
e = len(d)
f = e > a
g = d[0] == 's'
h = not (d[1] == 'i')
i = 'Up' + d
j = i < d
k = a // 2
l = a % 2
m = k < a
n = l == 1
p = random.randrange(5)
q = p != p
r = p == random.randrange(5)
s = r is None
```

Indicate the value of each of the variables below by ***circling*** exactly one of **T** (`True`), **F** (`False`) *NB* ("not Boolean" — i.e., the value is not of Boolean type) or *?* (cannot determine the answer with information provided). In the space to the right of your answer, briefly explain your reasoning.

var	*value*				*reason*
a	**T**	**F**	*NB*	*?*	
b	**T**	**F**	*NB*	*?*	
c	**T**	**F**	*NB*	*?*	
d	**T**	**F**	*NB*	*?*	
e	**T**	**F**	*NB*	*?*	
f	**T**	**F**	*NB*	*?*	
g	**T**	**F**	*NB*	*?*	
h	**T**	**F**	*NB*	*?*	
i	**T**	**F**	*NB*	*?*	
j	**T**	**F**	*NB*	*?*	
k	**T**	**F**	*NB*	*?*	
l	**T**	**F**	*NB*	*?*	
m	**T**	**F**	*NB*	*?*	
n	**T**	**F**	*NB*	*?*	
p	**T**	**F**	*NB*	*?*	
q	**T**	**F**	*NB*	*?*	
r	**T**	**F**	*NB*	*?*	
s	**T**	**F**	*NB*	*?*	

5 if/elif/else

Consider the following function:

```
def multiway(s):
    y = 63
    if (len(s) > 2) and (s[0] < s[1]) and (s[1] < s[2]):   # line A
        y -= 2
    elif (len(s) > 1) and (s[0] > s[1]):                   # line B
        y -= 4
    elif (len(s) < 4) or (s[0] == s[1]):                   # line C
        y -= 8
    else:
        y -= 16                                            # line D
    y = y // 2                                             # line E
    return y
```

a. Evaluate each of the following expressions (that is show what value the expression would return if evaluated in a Python program) and indicate which of the lines (**A** – **E**) would be executed on the way toward returning the final result.

`multiway('')`

`multiway('ACT')`

`multiway('zoo')`

`multiway('Peru')`

`multiway('oodles')`

---

b. What kinds of strings (in terms of length) cause `multiway` to return `23` when passed in as arguments? Give a specific example of a such a string.

c. Suppose line **C** was changed to read:

```
elif s[0] == s[1] or len(s) < 4:
```

Would that change the result of evaluating `multiway('9')`? If so, how? What, if anything, does that illustrate about the `or` operator?

6 arithmetic function

Consider the following function:

```
1  def co(n):
2      z = 0
3      m = n
4      while m > 0:          # <---
5          if m % 2 == 1:
6              z += 1
7          m = m // 2
8      return z
```

a. What does `co(0)` return?

b. What does `co(2)` return?

c. Suppose `co(13)` is evaluated. Indicate the values of `m` and `z` the first four times line 4 is executed.

#	m	z
1		
2		
3		
4		

d. Write a one-sentence description (that might be suitable for its docstring) of what function `co` returns.

7 string function

Consider the following function:

```
1  def hd(s):
2      found = False
3      if len(s) > 1:
4          i = 1
5          while not found and i < len(s):
6              found = (s[i] == s[i-1])
7              i += 1
8      return found              # <---
```

a. What is the type of variable `found`? How did you determine that?

b. What does `hd('x')` return?

c. Suppose `hd('cool?')` is evaluated. What is the value of `i` when line 8 is executed? What value does the function return?

d. Suppose `hd('Eek!')` is evaluated. What is the value of `i` when line 8 is executed? What value does the function return?

e. Write a one-sentence description (that might be suitable for its docstring) of what function `hd` returns.

8 comparing two functions

Consider the following pair of functions:

```
def looper(s):
    if len(s) % 2 == 0:
        half_length = len(s) // 2
        i = 0
        ok = True
        while ok and i < half_length:
            ok = (s[i] == s[i+half_length])
            i += 1
    else:
        ok = False
    return ok


def slicer(s):
    m = len(s) // 2
    return s[:m] == s[m:]
```

Convince yourself that they solve the same problem.

a. Write a one-sentence description (suitable for either function's docstring) of what they return.

b. Give a reason why `looper` might be considered better than `slicer`.

c. Give a reason why `slicer` might be considered better than `looper`.

d. Overall, which function do you prefer and why?

9 composition of functions

Suppose we have three functions `f`, `g`, and `h` where `f` and `g` are defined as:

```
def g(x):
    if x % 2 == 0:
        y = h(x, x)
    else:
        y = h(x, 2*x)
    return y

def f(u, v):
    z = 0
    i = u
    while i < v:
        if g(i):
            z += 1
        i += 1
    return z
```

and `h` is defined and runs without error, takes two arguments, both integers, always returns a Boolean value, *and*, if its arguments are equal, it returns `True`. (We have no idea what it returns if its arguments are not equal.) Suppose `f(1, 10)` is evaluated. What are the smallest and largest values that it might return? Explain your reasoning.

---

10 thought question

Give one (and only one) idea that you have learned in this course so far that you found particularly surprising or interesting. Explain briefly.