Partner #1: _____ Partner #2: _____

Homework 11: Memory

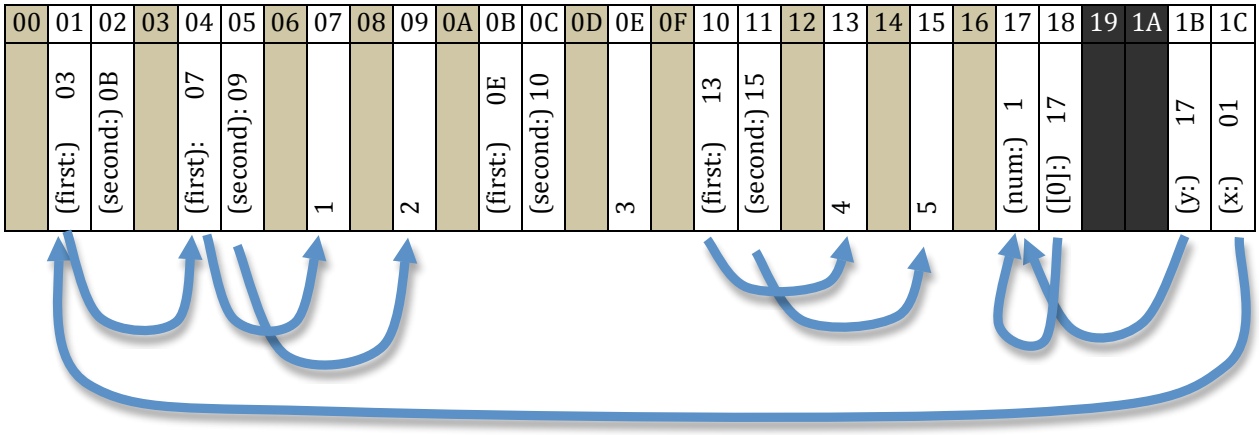
Due Thursday, May 7 at 9:55 am

As with the previous assignment, you may work with up to one other person only on this assignment. See HW10 for the rules.

Consider the evaluation of a Python program that begins with the three lines of code:

```
x = ((1,2), (3, (4, 5)))
y = [1]
y[0] = y
```

Executing the program up to this point produces the memory layout shown below it. For simplicity, assume that each memory location holds one byte, and that there are only 256 bytes in memory and integers can only have values on the range [0, 255]. The descriptions in parentheses are not actually stored in the program's memory—they are just there to help you know what variable. Note that I've printed all integers in hexadecimal, which is how memory dumps are usually displayed. The background shading is intended to help you recognize object (block) boundaries and is not stored in memory. The black blocks are unused. Assume that Python encodes all values as object instances on the heap, except for None (i.e., the Python null value).



1. Memory locations (4 points)
   a. At what address does the heap start, and which way does it grow?

   b. At what address does the stack start, and which way does it grow?

2. I've drawn in some of the pointers below the diagram as arrows from the variable that references an object to the start of the object in memory. Draw the remaining pointers. (9 points)

3. It is common in many interpreters and compilers to store the size of a block and its type right before the start of a block. Those memory locations are shown as shaded in the diagram. Assume that memory blocks may be no longer than 16 bytes, so that the size can be encoded in the high nibble[1] and the type in the low nibble. Assume that Python uses the type codes: integer = 0, array = 1, 2-tuple = 2. For example, an array of length 8 contains 9 fields: 1 array length and 9 pointers, so the block containing it might be represented as 91, since 9=size and 1=array. Fill in the shaded blocks with the appropriate values. (9 points)

[1] Nibble = 4 bits, i.e., ½ a byte

4. If the next two lines of the program are:

   x = None
   y = None

   then, immediately after executing those statements, which memory locations would be free under the following memory management schemes? Give your locations in hex and in increasing order; you can list inclusive ranges, e.g. 00-16. (9 points *be careful, this is a trick question!*)

   a. Manual memory management

   b. Reference counting

   c. Mark-sweep garbage collection

5. Assume that the initial three statements, and then the two statements from question 4 have executed. The next line is: a = 20. What would happen/which memory locations would then be free under the following memory management schemes? (9 points)

   a. Manual memory management

   b. Reference counting

   c. Mark-sweep garbage collection

6. Name two drawbacks of Mark-Sweep garbage collection compared to the (unachievable) ideal algorithm (5 points)

7. How long did the non-challenge part of this assignment take you? (2 points)

Challenge Question (no points)

8. What would the answers to questions 4 and 5 be if type flags were ignored and a conservative collector was used?