

Last Name: _____ First: _____ Netid: _____

CS 1110 Prelim 1 October 18th, 2020

This 90-minute exam has 5 questions worth a total of 100 points. Read over the whole test before starting. Budget your time wisely. Use the back of the pages if you need more space. You may tear the pages apart; we have a stapler at the front of the room.

It is a violation of the Academic Integrity Code to look at any exam other than your own, to look at any other reference material, or to otherwise give or receive unauthorized help.

You will be expected to write Python code on this exam. We recommend that you draw vertical lines to make your indentation clear, as follows:

```
def foo():  
    | if something:  
    |     | do something  
    |     | do more things  
    | do something last
```

You should not use loops or recursion on this exam. Beyond that, you may use any Python feature that you have learned in class (if-statements, try-except, lists), **unless directed otherwise**.

Question	Points	Score
1	2	
2	23	
3	26	
4	24	
5	25	
Total:	100	

The Important First Question:

1. [2 points] Write your last name, first name, and netid, at the top of *each* page.

Reference Sheet

Throughout this exam you will be asked questions about strings and lists. You are expected to understand how slicing works. In addition, the following functions and methods may be useful.

String Functions and Methods

Expression or Method	Description
<code>len(s)</code>	Returns: number of characters in <code>s</code> ; it can be 0.
<code>a in s</code>	Returns: True if the substring <code>a</code> is in <code>s</code> ; False otherwise.
<code>s.count(s1)</code>	Returns: the number of times <code>s1</code> occurs in <code>s</code>
<code>s.find(s1)</code>	Returns: index of the first character of the FIRST occurrence of <code>s1</code> in <code>s</code> (-1 if <code>s1</code> does not occur in <code>s</code>).
<code>s.find(s1,n)</code>	Returns: index of the first character of the first occurrence of <code>s1</code> in <code>s</code> STARTING at position <code>n</code> . (-1 if <code>s1</code> does not occur in <code>s</code> from this position).
<code>s.isalpha()</code>	Returns: True if <code>s</code> is <i>not empty</i> and its elements are all letters; it returns False otherwise.
<code>s.isdigit()</code>	Returns: True if <code>s</code> is <i>not empty</i> and its elements are all numbers; it returns False otherwise.
<code>s.isalnum()</code>	Returns: True if <code>s</code> is <i>not empty</i> and its elements are all letters or numbers; it returns False otherwise.
<code>s.islower()</code>	Returns: True if <code>s</code> is <i>has at least one letter</i> and all letters are lower case; it returns False otherwise (e.g. <code>'a123'</code> is True but <code>'123'</code> is False).
<code>s.isupper()</code>	Returns: True if <code>s</code> is <i>has at least one letter</i> and all letters are upper case; it returns False otherwise (e.g. <code>'A123'</code> is True but <code>'123'</code> is False).

List Functions and Methods

Expression or Method	Description
<code>len(x)</code>	Returns: number of elements in list <code>x</code> ; it can be 0.
<code>y in x</code>	Returns: True if <code>y</code> is in list <code>x</code> ; False otherwise.
<code>x.count(y)</code>	Returns: the number of times <code>y</code> occurs in <code>x</code>
<code>x.index(y)</code>	Returns: index of the FIRST occurrence of <code>y</code> in <code>x</code> (an error occurs if <code>y</code> does not occur in <code>x</code>).
<code>x.index(y,n)</code>	Returns: index of the first occurrence of <code>y</code> in <code>x</code> STARTING at position <code>n</code> (an error occurs if <code>y</code> does not occur in <code>x</code>).
<code>x.append(y)</code>	Adds <code>y</code> to the end of list <code>x</code> .
<code>x.insert(i,y)</code>	Inserts <code>y</code> at position <code>i</code> in list <code>x</code> , shifting later elements to the right.
<code>x.remove(y)</code>	Removes the first item from the list whose value is <code>y</code> (an error occurs if <code>y</code> does not occur in <code>x</code>).

The last three list methods are all procedures. They return the value `None`.

2. [23 points total] **Objects and Functions.**

Remember the class `RGB` from Assignment 3. Objects of this class have three attributes: `red`, `green`, and `blue` (we will ignore `alpha` for this question). These values must be integers between 0 and 255; assigning any other value to them will result in an error.

- (a) [9 points] When we multiply two colors together, we do it attribute by attribute. That is, we first convert the attributes to the range 0..1 (as in the assignment) and then multiply each attribute like this

$$C' = C_1 * C_2 \quad \text{Multiply the color attributes } C_1 \text{ and } C_2$$

where C_i is **each one of red, green, or blue**. We then convert each value to the range 0..255 when done (remembering to round the result). With that in mind, implement the function below.

NOTE: You do not need to worry about importing `intros`. Assume it is available.

```
def multiply(color1,color2):
    """Returns a new RGB object that is the product of color1 and color2

    The original colors should not be modified

    Preconditions: color1 and color2 are RGB objects"""
```

Last Name: _____ First: _____ Netid: _____

- (b) [14 points] The opposite of multiplication is division. To divide two colors, we again convert the attributes to the range 0..1 (as in the assignment) and then use the formula

$$C' = \begin{cases} 0 & \text{if } C_1 = 0 \\ C_1/C_2 & \text{if } C_1/C_2 \leq 1 \\ 1 & \text{otherwise} \end{cases}$$

where C_i is **each one of red, green, or blue**, converting back to the range 0..255 when done (remembering to round the result). Using this formula, implement the function below.

Note: In mathematics, $a/0$ is ∞ whenever $a > 0$. That is implicit in the formula above.

```
def divide(color1,color2):
    """MODIFIES color1 to store the result of division.

    This function divides color1 by color2 and stores the result in color1.

    Preconditions: color1 and color2 are RGB objects"""
```

3. [26 points total] **Testing and Debugging.**

(a) [10 points] The function `romanize` takes a integer 1..99 and converts it into a Roman numeral such as I, VII, or XIX. Roman numerals are represented as strings with the letters 'I', 'V', 'L', 'X', and 'C'.

There are *at least three bugs* in the code below. These bugs are potentially spread across multiple functions. To help find the bugs, we have added several print statements throughout the code, and show the results on the next page. Using this information as a guide, identify and fix the three bugs on the next page. Your fixes may include more than one line of code. You should explain your fixes.

```

1  def romanize(n):
2      """Returns the Roman numeral for n
3
4      Precond: 0 < n < 100 is an int"""
5      tens = ''
6      ones = ''
7      if n >= 50:
8          print('More than 50')      # TRACE
9          tens = numeralL(n//10)
10     elif n >= 10:
11         print('More than 10')      # TRACE
12         tens = numeralX(n//10)
13     print('tens = '+repr(tens))    # WATCH
14
15     ones = romanize1to9(n % 10)
16     print('ones = '+repr(ones))    # WATCH
17
18     def romanize1to9(n):
19         """Returns the Roman numeral for n
20
21         Precond: 0 < n < 10 is an int"""
22         # Combined TRACE and WATCH
23         print('romanize1to9: n = '+repr(n))
24         if n < 5:
25             print('Less than 5')    # TRACE
26             return romanize1to4(n)
27         elif n < 9:
28             print('Between 5 and 8') # TRACE
29             return 'V'+romanize1to4(n-5)
30         else:
31             print('Equal to 9')      # TRACE
32             return 'IX'
33
34     def romanize1to4(n):
35         """Returns the Roman numeral for n
36
37         Precond: 0 < n < 5 is an int"""
38         # Combined TRACE and WATCH
39         print('romanize1to4: n = '+repr(n))
40         values = ['I','II','III','IV']
41         choiceI = values[n-1]
42         # WATCH
43         print('choiceI = '+repr(choiceI))
44         return choiceI
45
46
47
48     def numeralL(n):
49         """Returns Roman numeral for tens value
50
51         The value n is the tens DIGIT of the
52         number. So numeralL(5) is 'L'.
53
54         Precond: 5 <= n < 10 is an int"""
55         # Combined TRACE and WATCH
56         print('numeralL: n = '+repr(n))
57         if n < 9:
58             print('Less than 90')    # TRACE
59             return 'L'+numeralX(n-5)
60         else:
61             print('Equals to 90')    # TRACE
62             return 'XC'
63
64
65
66
67
68     def numeralX(n):
69         """Returns Roman numeral for tens value
70
71         The value n is the tens DIGIT of the
72         number. So numeralL(3) is 'XXX'.
73
74         When n is 0, it returns the empty
75         string (to be compatible w/ numeralL)
76
77         Precond: 0 <= n < 5 is an int"""
78         # Combined TRACE and WATCH
79         print('numeralX: n = '+repr(n))
80         values = ['', 'X', 'XX', 'XXX', 'XL']
81
82         choiceX = values[n]
83         # WATCH
84         print('choiceX = '+repr(choiceX))
85         return choiceX
86
87
88

```

Last Name: _____ First: _____ Netid: _____

Hint: Some bugs cannot be fixed with just one line. You might need to add a conditional.

Tests:

```
>>> romanize(14) # Expected: 'XIV'
More than 10
numeralX: n = 1
choiceX = 'X'
tens = ''
romanize1to9: n = 4
Less than 5
romanize1to4: n = 4
choiceI = 'IV'
ones = 'IV'
IV
```

First Bug:

```
>>> romanize(75) # Expected: 'LXXV'
More than 50
numeralL: n = 7
Less than 90
numeralX: n = 2
choiceX = 'XX'
tens = 'LXX'
romanize1to9: n = 5
Between 5 and 8
romanize1to4: n = 0
choiceI = 'IV'
ones = 'VIV'
LXXVIV
```

Second Bug:

```
>>> romanize(60) # Expected: 'LX'
More than 50
numeralL: n = 6
Less than 90
numeralX: n = 1
choiceX = 'X'
tens = 'LX'
romanize1to9: n = 0
Less than 5
romanize1to4: n = 0
choiceI = 'IV'
ones = 'IV'
LXIV
```

Third Bug:

Last Name: _____ First: _____ Netid: _____

(b) [10 points] Consider the specification below

```
def count_adjacent(a,b):  
    """Returns the number of adjacent pairs of b inside of the string a.  
  
    An adjacent pair just means that two copies of b appear next to each other  
    in a. For example, if b = 'b' it appears twice as an adjacent pair in the  
    string 'abbcbbdb'.  
  
    Preconditions: a and b are both nonempty strings of lowercase letters.
```

Do not implement this function. Instead, we want you to write at least **six test cases** below. By a test case, we just mean an input and an expected output; you do not need to write an `assert_equals` statement. For each test case, you should explain why it is substantially different from the others.

(c) [6 points] **Do not implement the function specified below.** Instead, use assert statements to enforce the precondition. Furthermore, each the assert statement should produce one of the three error messages shown below

```
>>> after_space(13)
AssertionError: 13 is not a string.
>>> after_space('abc')
AssertionError: 'abc' has no spaces.
>>> after_space(' abc')
AssertionError: ' abc' has an illegal space.
```

```
def after_space(s):
    """Returns the part of the string after the first space in s

    Precond: s a string with at least one space.
    Furthermore, s does not start or end with a space."""
```

4. [24 points] **Call Frames.**

Consider the following function definitions.

```
1 def fold_front(p):
2     """Returns sum of first 2 items
3     This function modifies the list
4     Pre: p is a list, len(p) >= 2"""
5     p[0]= p[0]+p[1]
6     return p[0]
7
8 def sum_back(q):
9     """Returns sum of last 2 items
10    Pre: q is a list, len(q) >= 2"""
11    r = q[-2:]
12    result = fold_front(r)
13    return result
14
```

Assume `p = [3, 7, 1, 5]` is a global variable referencing a list in the heap, as shown on the next page. On the next two pages, diagram the evolution of the call

```
r = sum_back(p)
```

Diagram the state of the *entire call stack* for the function `sum_back` when it starts, for each line executed, and when the frame is erased. If any other functions are called, you should do this for them as well (at the appropriate time). This will require a total of **eight** diagrams, not including the (pre-call) diagram shown.

You should draw also the state of global space and the heap at each step. You can ignore the folders for the function definitions. Only draw folders for lists or objects. You are also allowed to write “unchanged” if no changes were made to either global space or the heap.

Last Name: _____

First: _____

Netid: _____

Call Stack

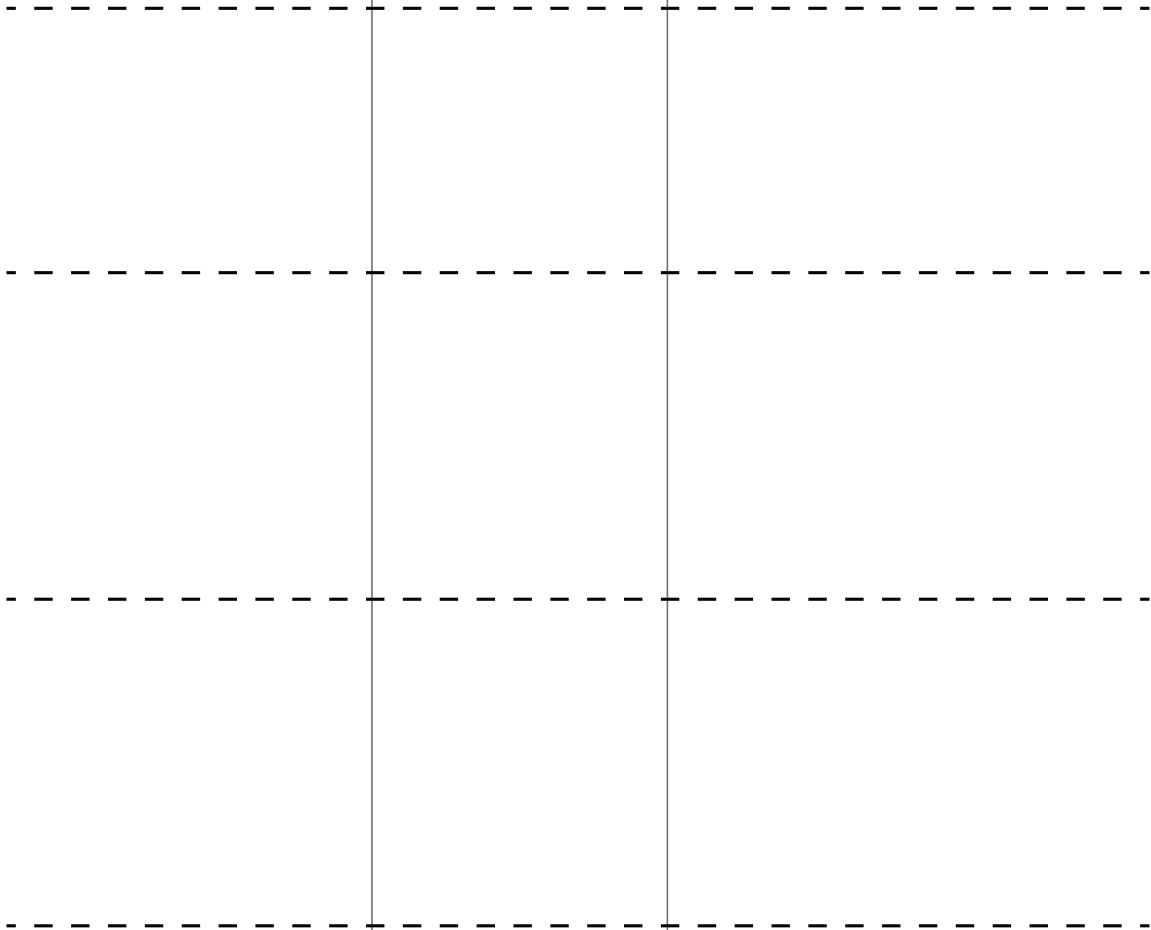
Global Space

The Heap

p

id1

id1	
	list
0	3
1	7
2	1
3	5



Last Name: _____

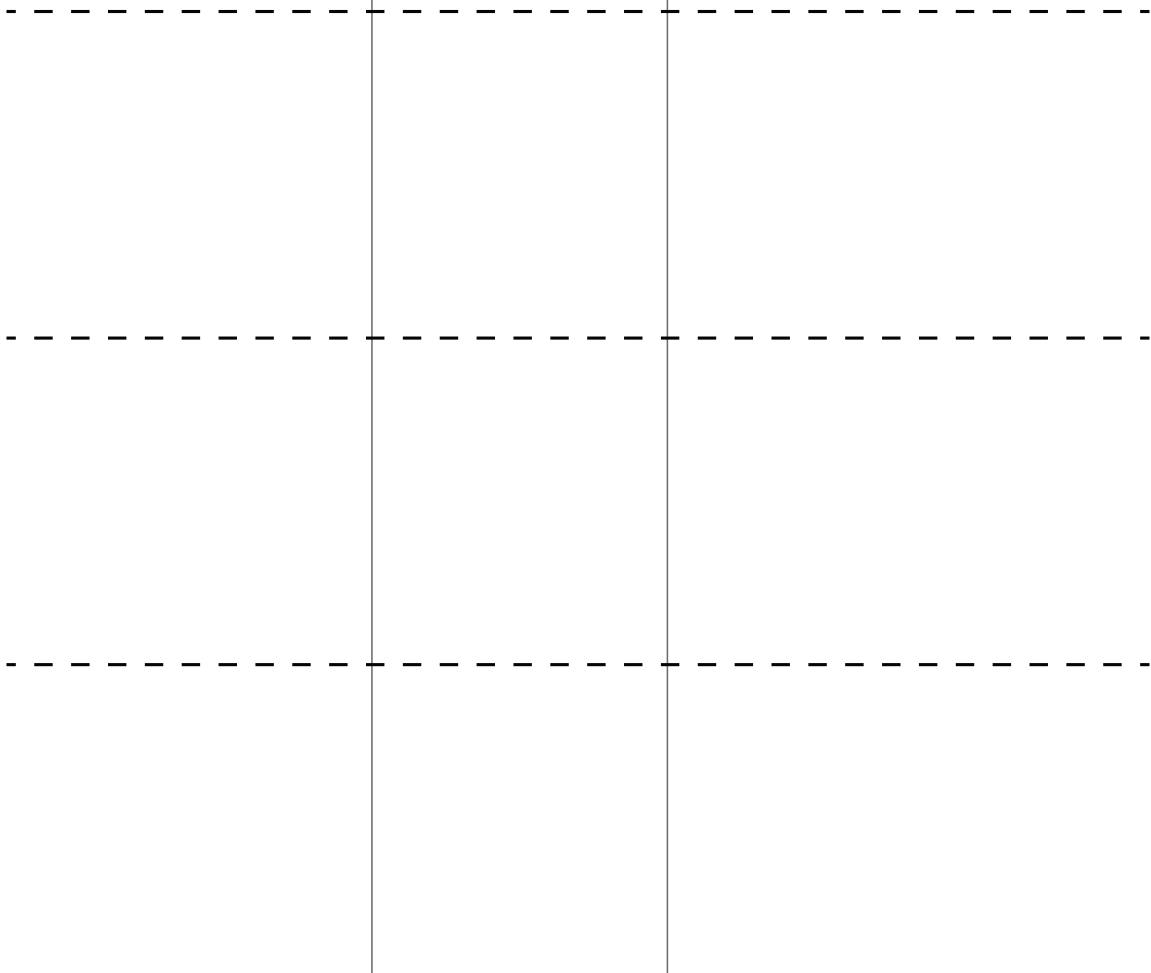
First: _____

Netid: _____

Call Stack

Global Space

The Heap



Last Name: _____ First: _____ Netid: _____

5. [25 points] **String Slicing.**

Implement the function below. You may need to use several of the functions and methods on the reference page. Pay close attention to the examples to better understand the function.

```
def swap_first(word,a,b):
    """Returns a copy of word with first instance each of a and b swapped

    If either a or b is not a substring of word, then word is unchanged.
    If a and b overlap inside of word, then the word is also unchanged.

    Examples:
        swap_first('aBcD','B','c') returns 'acBD'
        swap_first('aBcD','c','B') returns 'acBD'
        swap_first('aBcD','x','c') returns 'aBcD'
        swap_first('aBCdeF','BC','de') returns 'adeBCF'
        swap_first('aBCdeF','BC','Cd') returns 'aBCdeF'

    Preconditions: word, a, and b are all non-empty strings"""
```