|  |
| --- |
| 15-110 Recitation 04 |

**Recap**

* Lists (append, indexing, len, max, sum, splicing, iterating)
* 2D lists
* List aliasing

**Reminders for Students**

* Check 3 due Monday 10/5 @ Noon EDT
* Check 2 and HW 2 revisions due Tuesday 10/6 @ Noon EDT
* Test 2 is Wednesday 10/7, same procedures as Test 1
	+ These “tests” are like quizzes, lower stakes & shorter in length, but more frequent
	+ Don’t stress out about these too much!

|  |
| --- |
| Problems |

# **DEBUG LISTS**

**Identify the bug in the following functions and describe how to fix them.**

1. The following function returns a string that is the concatenation of all the strings in a list

def f1(string\_list):

 big\_string = ""

 for string\_elem in range(0,len(string\_list)):

 big\_string = big\_string + string\_elem

 return big\_string

Bug:

Fix:

1. The following function returns a list that contains all the non-negative integers less than n that are a multiple of k

def f2(n, k):

 multiple\_list = []

 for i in range(0, n):

 if (i % k == 0):

 multiple\_list.append([i])

 return multiple\_list

Bug:

Fix:

# **2D LISTS RUNDOWN**

**Finding dimensions:**

a = [ [ 2, 3, 5] , [ 1, 4, 7 ] ]

What is a?

How many rows are in a?

How many columns are in a?

**Looping through each element (increases each element by 1 for example):**

a = [ [ 2, 3, 5] , [ 1, 4, 7 ] ]

# dimensions

rows = len(a)

cols = len(a[0])

for row in range(rows):

 for col in range(cols):

 a[row][col] += 1 #<- how many times will this line be run?

What is a after running this code?

# **LIST ALIASING**

Code trace and compare the following two options for ways to create “empty” 2D lists in Pyzo:

Option 1:

rows = 3

cols = 2

a = [ [0] \* cols ] \* rows

Option 2:

def make2dList(rows, cols, defaultValue=None):

 a=[]

 for row in range(rows):

 a.append([defaultValue]\*cols)

 return a

rows = 3

cols = 2

a = make2dList(rows, cols, 0)

For each option, after running the code, what is a?

After adding the following line of code:

a[0][0] = 42

What is a?

Be sure you can explain what difference you are seeing, and which option you should use and why.

# **ALTERNATING SUM**

Write a function alternatingSum(L) that takes in a list of numbers L, and then returns the alternating sum (where the sign switches from positive to negative or negative to positive at each index).

For example, alternatingSum([5,3,8,4]) returns 6 because (5-3+8-4) = 6

See starter file for more tests and function header!

def

# **MULTIPLICATION TABLE**

Write a function createMultiplicationTable(n) that takes in a number n, and then returns an nxn multiplication table where the ith row and jth column represents the result of (i+1)\*(j+1).

createMultiplicationTable(3) -> [[1,2,3],[2,4,6],[3,6,9]]

Remember you can think of this as a table:

 [ [1,2,3],

 [2,4,6],

 [3,6,9] ]

def