15-112 Fundamentals of Programming
Practice Midterm I
Summer I 2017
80 minutes

Name : _____

Andrew Id : _____**@andrew.cmu.edu**

Section : _____

- **You may not use any books, notes, or electronic devices during this exam.**
- **You may not ask questions about the exam except for language clarifications.**
- **Show your work on the exam (not scratch paper) to receive credit.**
- **If you use scratch paper, you must submit it with your andrew id on it, and we will ignore it.**
- **All code samples run without crashing.**
- **Assume any imports are already included as required**

DO NOT WRITE IN THIS AREA

Part 1 : Code Tracing (15)	
Part 2 : Reasoning Over Code (15)	
Part 3 : Short Answers (10)	
Part 4 : Big Oh Oh (15)	
Part 5 : (FR) nthAlternatingNumber (15)	
Part 6 : (FR) funkyTable (15)	
Part 7 : (FR) switch n flip (15)	
Part 8 : Bonus (5)	

Part 1 : Code Tracing (15)

import copy
def ct1(a):
  L, M, N = a, a[::], copy.deepcopy(a)
  L[0][0] += 2
  M[1] += [3]
  M[1] = M[1] + [42]
  N[1] = L[0]
  N[1][0] = N[1][0] + 2
  return (M, N)
 	
L = [[1],[]]
M, N = ct1(L)
for A in [L,M,N]: print(A)
print(L)	

import string
def ct2(s, t):
  r = ''
  for c in s.lower() :
  if c.upper() in t :
  r += str(t.find(c.upper()))
  print(c, end = " ")
  else :
  t = t[: :-1]
  print (r, s, t)
 	
print(ct2('yasOnetwelve', 'EVLEWTENO'))	

def ct3(n):
  count = 0
  m = n
  while n > 0 :
  count += 1
  n //= 10
  N = count
  L = [i + 1 for i in range(N)]
  L2 = [L[i]**N for i in range(N)]
  return (L2, L)
 	
print(ct3(1234))	

Part 2 : Reasoning Over Code (10)

def rc1(L):
  r = s = 0
  assert(len(L) > 1)
  for n in L:
  if (n > 4000): return False
  while (n > 0):
  (d, n) = (n%10, n//10)
  if (d % 2 == 0):
  r = 10*r + d
  else:
  s = 10*s + d
  return (r,s) == (68, 53)	

def rc2(s):
  If (not type(s) == str): return False
  T = string.ascii_uppercase
  return ((s[3::3] == t[0:3]) and (s[-2::-2] == t[:5])
  and (s.count(s[0]) > 4))	

Part 3 : Short Answer (15)

1. Prove the worst case run time of bubble sort, selection sort and merge sort.

2. Assuming selectionsort sorts largest numbers first, give an example of a list where bubblesort and selectionsort would each make the same 3 swaps in the same order to fully sort the list.

3. In solving wordSearch, we wrote a main function which called a second function which in turn called a third function. What was the main purpose of the second (not the outermost, and not the innermost) function we wrote?

4. Give 2 examples of short-circuit evaluation.

5. TRUE of FALSE: If ((x == y) != (x is y)) is False, then x is an alias of y.

Part 4 : Big-Oh (15)

def lalala_01(s) : # s is a string of len N
r = ""
for c in s :
r += c
return r	

def lol(L): # L is a 1D list if length N
M = [L[i] * L[j] for i in range(len(L)) for j in range(len(L))]
K = sorted(M)
for elem in K :
if elem == 1 :
print("yas I am doing well today")
else :
print(" lol wtf is happening")
return	

def OneTwelve(n): # n is an int for i in range(112, n, 112) : print("Welcome to the dark side.", end = "\t") n += n return True # give answer in terms of N, not n	

Part 5 : (FR) nthAlternatingNumber(n) (15)

You should be able to complete this problem without using lists or strings. If you use strings / lists you **automatically lose half the points**.

An integer n is an Alternating Number (a coined term) if and only if:

1. n is positive
2. n has at least 4 digits
3. n contains only *2 unique digits*
4. None of the consecutive digits in n are the same value

For example, if n == 62626:

- n has greater than four digits
- The only unique digits in n are 2 and 6
- There are no runs of 2 or 6 greater than one

So 62626 is an Alternating Number.

The first several Alternating Numbers are:
1010, 1212, 1313, 1414, 1515, 1616, 1717, 1818, 1919, 2020, 2121, 2323…

With this in mind, write the function nthAlternatingNumber(n), which takes a non-negative integer and returns the nth Alternating Number.

Part 6 : (FR) funkyTable(canvas, width, height, cellSize) (15)

We've seen how to create tables in tkinter in class, but those are so boring and plain, so in this question we're going to spice things up! Write the function funkyTable that takes the following parameters:

- Canvas
- Width and height of the window
- cellSize: the height and width of each cell

Your function should first create a basic table of square cells that contains the maximum possible number of rows and columns that you can draw on the window without going off the edge of the screen. Within each cell you should then draw a circle, and then on top of that circle a diamond. Cells should display the text "42" and "112" with each number being placed in an alternating pattern across the table. To really make this funky, each level of the drawing (the squares, circles, diamonds, and numbers) should each have a different color.

You may assume the runDrawing() function is already defined for you.

PART 7 : (FR) switchLines(L, type, a0, a1) and flipList(L) (15)

For this question, first you will define the function switchLines, which takes a rectangular 2D list L and will return a new list with either two rows or two columns in the original list switched. The type parameter will specify whether you should be looking at a row or column (you may assume the only values that you will see for this parameter are the strings “row” and “col”). The integers a0 and a1 will specify which rows or columns you should switch.

For example, if L ==
[[1, 2],
  [3, 4]]

switchLines(L, “row”, 0, 1) should return:

[[3, 4],
  [1, 2]]

If either a0 or a1 is out of bounds for the list, return the original list.

Once you have completed this, write the function flipList(L), using switchLines as a helper function, which takes a rectangular 2D list and returns a list which is in the reversed order of the original (i.e. the first column and the last column have been switched, the second and second last columns have switched, etc. and the same is true for all the rows).

Bonus (5)

def bonusCT1(): # 2.5 points a = [int(math.log(x,3)) for x in range(1,3)] for k in range(1234): a = [a[0] or k, a[1] and k] return int("".join([str(x)*x**y for x in a for y in a])) print(bonusCT1())	
def bonusCT2(L): # 2.5 points def f(L):return L[:]*len(L[:]) + [val*len(L) for val in L] def g(L, n): return len(L) if (len(L) >= n) else g(f(L), n) return g(L, 42) print(bonusCT2(list(range(5))))	