15-112 Midterm #1 – Spring 2015
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.**
- **#5 is optional. You may skip it if you wish without penalty.**

DO NOT WRITE IN THIS AREA		
Part 1 (CT)	10 points	
Part 2 (RC)	10 points	
Part 3 (SA)	15 points	
Part 4 (FitB)	10 points	
Part 5 (FR / Optional)	10 points	
Part 6 (FR)	15 points	
Part 7 (FR)	10 points	
Part 8 (FR)	20 points	
Part 9/bonus	5 points bonus	
Total	100 points	

1. **[10 pts] Code Tracing. Indicate what each will print:**

Statement(s):	Prints:

`def f(a):`
`  s = "f"`
`  for row in a: s += str(row[0])`
`  return s`
 	
`def ct1(a):`
`  b = copy.deepcopy(a)`
`  c = a+a`
`  b[0][0] = 3`
`  c[0][0] = 4`
`  b[1] = [5]`
`  c[1] = [6]`
`  print f(b), f(c) # f is defined above`
 	
`a = [[1],[2]]`
`ct1(a)`
`print f(a) # don't miss this`
	_____
	
	_____

`def ct2(a):`
`  s = set(a)`
`  t = copy.copy(s)`
`  for val in s:`
`  if (val%3 != 0): t.remove(val)`
`  print sorted(s)`
`  print sorted(s - t)`
 	
`ct2([(n**2)%10 for n in xrange(3,8)])`
	_____
	
	_____

2. **[10 pts] Reasoning about code**
For each of the functions f, find values of the parameters so that f will return True. Place your answers in the box on the right of each function.

`def rc1(a):`
`  s = ""`
`  for t in a: s += t`
`  assert(s == string.ascii_lowercase[2:9])`
`  return ((len(a) == len(a[1])) and`
`  (len(a) == ord(s[2]) - ord('a')))`
	a = _____

`def rc2(n):`
`  d = 9`
`  while (d > 2):`
`  if (n%10 != d): return False`
`  n /= 10`
`  d = 1 + d/2`
`  return (n == 42)`
	n = _____

3. **[15 pts] Short Answer:** Answer each of the following very briefly.
This problem concerns the following code. Note that your solutions may use any built-in functions, methods, or data structures we have covered this semester.

```
##############################################
def swap(a, i, j):
    (a[i], a[j]) = (a[j], a[i])

def slow(L):
    # Assume L is a non-empty list of integer values
    a = copy.copy(L)
    # the following code (until the "end quote") is quoted verbatim from
    # a case study in the course notes
    n = len(a)
    for startIndex in xrange(n):
        minIndex = startIndex
        for i in xrange(startIndex, n):
            if (a[i] < a[minIndex]):
                minIndex = i
        swap(a, startIndex, minIndex)
    # end quote
    for i in xrange(len(a)):
        if (a[i] != i): return False
    return True
##############################################
```

A. The quoted code is taken from which function in the course notes?

B. State and <u>briefly</u> prove the worst-case big-oh of slow(L), assuming L is a list of n integers.

This question continues on the next page

Reminder: your solutions may use any built-in functions, methods, or data structures we have covered this semester.

C. Write faster(L) that works identically to slow(L) but runs in O(nlogn), though not in O(n).

D. Write fastest(L) that works identically to slow(L) but is even faster than faster(L), running in O(n).

4. **[10 pts] Fill in the Blanks: rectOf5(L)**

Background: The following function, rectOf5(), is mostly written for you, in fact in two different ways (both correct). Your task is to fill in the missing blanks. First, we will describe the function.

The function rectOf5(L) that takes a rectangular 2d list L that is guaranteed to have at least one row and at least one col, and returns True if it contains at least one 5 and all the 5's in L form a full rectangle, and False otherwise. For example:

```
L = [ [  1, 2, 1, 2, 1 ],
      [  3, 5, 5, 5, 3 ],
      [  2, 5, 5, 5, 1 ],
      [  1, 2, 2, 3, 4 ]
    ]
```

Here, we see a full 2x3 rectangle of 5's, and no other 5's, so the function would return True for this list. Another example:

```
L = [ [  1, 2, 1, 2, 1 ],
      [  3, 5, 0, 5, 3 ],
      [  2, 5, 5, 5, 1 ],
      [  1, 2, 2, 3, 4 ]
    ]
```

Here, there is a 0 inside the rectangle of 5's, so the function would return False. One more example:

```
L = [ [  5, 2, 1, 2, 1 ],
      [  3, 5, 5, 5, 3 ],
      [  2, 5, 5, 5, 1 ],
      [  1, 2, 2, 3, 4 ]
    ]
```

Here, there is a 5 outside the rectangle, so again the function would return False.

4a) testRectOf5()

This is a partially completed test function for this problem. Fill in the missing line of code with a single Python statement so that this works as expected for a test function:

```
def testRectOf5():
    print "Testing rectOf5()...",
    L = [ [  1, 2, 1, 2 ],
          [  3, 5, 5, 3 ] ]

    ____________________________________________________________
    print "Passed!"
```

This question continues on the next page

4b) rectOf5(L)

This is the first correct version of rectOf5(L). Fill in the blanks with short, accurate comments. Your comments must be at a high level, and not just detail what each line does at a low level. Be brief.

```
def rectOf5(L):
    (rows, cols) = (len(L), len(L[0]))
    seen5 = False

    # 1 of 3: Comment explaining
    # what the next block of code
    # is doing at a high level,
    # and in just a few words:   ____________________________________

    for row in xrange(rows):
        for col in xrange(cols):
            if (L[row][col] == 5):
                if (seen5 == False):
                    (row0, col0) = (row, col)
                    seen5 = True
                (row1, col1) = (row, col)

    if (seen5 == False):
        # 2 of 3: Comment explaining
        # what is true about L in
        # order to get here:     ____________________________________

        return False

    # 3 of 3: Comment explaining
    # what the next block of code
    # is doing at a high level,
    # and in just a few words:   ____________________________________

    for row in xrange(row0, row1+1):
        for col in xrange(col0, col1+1):
            if (L[row][col] != 5):
                return False
    return True
```

This question continues on the next page

4c) alternate rectOf5(L)

This is the first correct version of rectOf5(L). Fill in the blank with a single Python expression (not a statement), so the function works properly.

```
def rectOf5(L):
    # All the code below here is unchanged until the next comment
    (rows, cols) = (len(L), len(L[0]))
    seen5 = False
    for row in xrange(rows):
        for col in xrange(cols):
            if (L[row][col] == 5):
                if (seen5 == False):
                    (row0, col0) = (row, col)
                    seen5 = True
                (row1, col1) = (row, col)
    if (seen5 == False):
        return False
    # All the code above here is unchanged.

    for row in xrange(row0, row1+1):

        # The next line has 4 blanks for you to fill in

        if (L[row][ _______ : ____________ ] != [__]*( _____________ )):
            return False
    return True
```

5. **[10 pts] OPTIONAL Free Response: nthTennish(n)**

Note: This is the note we posted to piazza before the exam regarding the optional problem. This problem is optional. If you do it, and score over 5/10 on it, then it counts. But you may skip it entirely, making your raw score out of 90, which will be multiplied by 100/90 so your overall score remains out of 100 either way. We suggest you skip it now and return to it later if you have time.

We will say that a positive integer is *tennish* (a coined term) if its digits sum to 10. So, 127, 721, and 100200007 are all tennish. With this in mind, write the function nthTennish(n) that takes a non-negative int n and returns the nth tennish number. Note that nthTennish(0) returns 19, which is the smallest tennish number.

6. **[15 pts] Free Response: isShortestPath(path)**

We will say that a string is a *path* if it only contains the letters U, D, L, or R, which indicate to take one step in the directions up, down, left, and right, respectively. Two paths are *equivalent* if, when starting at (0,0) and following those paths, you end up at the same point. For example, the path "URUL" goes from (0,0) up to (0,1), right to (1,1), up to (1,2), and left to (0,2). The path "UU" goes from (0,0) up to (0,1) and up to (0,2). So "URUL" and "UU" are equivalent paths, as both end at (0,2). Also, note that it is fine for paths to move through negative indexes. With this in mind, write the function isShortestPath(path) that takes a string that is guaranteed to be a (possibly-empty) path, and returns True if it is shortest (so there is no shorter equivalent path) and False otherwise.

7. [10 pts] Free Response: stateList(d)

Background: for this problem, we will represent US states using their 2-letter state names ("FL" for Florida, "GA" for Georgia, etc). We will start with a dictionary d that maps each state to a set of their neighboring states. So d["FL"] is set(["GA", "AL"]), since Georgia (GA) and Alabama (AL) are Florida's only neighboring states. With this in mind, write the function stateLists(d) that takes a dictionary d as just described and returns a ragged 2d list L, so that L[n] is an alphabetically sorted list of every state with exactly n neighbors. L should not be any larger than required in either dimension. Thus, for example, L[7] is the sorted list ["CO", "KY"] because Colorado (CO) and Kentucky (KY) are the only states with exactly 7 neighbors. Note that in theory it is possible that L[k] may be the empty list, [ ], if there are no states with k neighbors, but there are states with more than k neighbors (though in reality this does not happen). Also note that you may not hardcode to the US map, so your function should work with a map of any states in any country.

8. **[20 pts] Free Response: event-based animation**

For this problem, you are given the following code:

```
def go(width, height, squareSize):
    eventBasedAnimation.run(initFn = initFn, stepFn = stepFn,
                            drawFn = drawFn,
                            mouseFn = mouseFn, keyFn = keyFn,
                            width=width, height=height,
                            squareSize=squareSize)
go(100, 100, 20) # do not hardcode to these values!
```

Using this code, write the functions initFn, stepFn, etc, to do the following:

1. Have a small red square of the given size (so, 20x20 in the example above) repeatedly sweep first from the bottom-left to the top-right corner of the window. When it gets to the top-right corner, it then proceeds straight down the right edge of the window. Each time it hits the bottom-right corner, it immediately starts over, jumping back to the bottom-left corner and sweeping up to the right again.

2. Display a score in the middle, horizontally, positioned near the top of the window. The score is initially 5 (not 0).

3. Each time the user clicks the mouse, if it is in the moving square, their score increases by 1. If it is outside the square, their score decreases by 1.

4. If the score gets to 10, the game is over, and the screen displays "You win!"

5. If the score gets to 0, the game is over, and the screen displays "You lose!"

6. When the game is over, the red box is not seen, and mouse presses are ignored.

7. When the game is over, and only then, the user can press "n" to start a new game, and the game starts over in its initial state.

[this page is blank (for your event-based animation)]

[most of this page is blank (for your event-based animation)]

Bonus/Optional: **[2.5 pts] What will this print?**

```
def bonus1(n):
    a = [(n+2)/(n+1)]*(n+0)
    for b in xrange(a[1]+a[-1],n):
        for c in xrange(b+b,n,b):
            a[c] = a[c]/(2+a[2])
    return int("".join(str(b) for b in a[2:]),2)
print bonus1(10)
```

Bonus/Optional: **[2.5 pts] What will this print?**

```
def bonus2(n):
    s = set()
    def f(n): s.add(n); return 0 if (n <= 1) else 1 + f(n/3)
    def g(n): return 0 if (f(n) <= 1) else 2**g(n/2)
    s.add(g(n)**2)
    return sorted(s)
print bonus2(11)
```