15-112 Fundamentals of Programming
Practice Midterm II
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	10 points	
Part 2 : Reasoning Over Code	5 points	
Part 3 : Short Answers	12 points	
Part 4 : (FR) stateList(d)	15 points	
Part 5 : (FR) parenContent	15 points	
Part 6 : (FR) TA and Kosbie classes	20 points	
Part 7 : (FR) playCircleGame	23 points	
Total	100 points	

1. Code Tracing [10 points]

Statement(s):	**Prints**

`def ct1(n, m = 0, depth = 0):`
`  print("   "*depth, "ct1(%d, %d)" % (n, m))`
`  if(m > n):`
`  result = m + n`
`  print("   "*depth, "-->", result)`
`  return result`
`  elif(m == n):`
`  result = ct1(n, m-1, depth+1)`
`  print("   "*depth, "-->", result)`
`  return result`
`  else:`
`  result = ct1(n//10, m+1, depth+1) + ct1(n-5, m+10, depth+1)`
`  print("   "*depth, "-->", result)`
`  return result`
 	
`#prints 9 lines`
`print(ct1(10, 10))`	

`def f(a):`
`  s = set()`
`  d = dict()`
`  for i in range(len(a)):`
`  if (i%2 == 0):`
`  if (a[i] in d):`
`  s.add(a[i+1])`
`  else:`
`  d[a[i]] = a[i+1]`
`  return (sorted(s), d[42])`
 	
`print(f([42,6,0,4,1,5,0,4,1,5]))`	

2. Reasoning Over Code [5 points]

Statements	Input

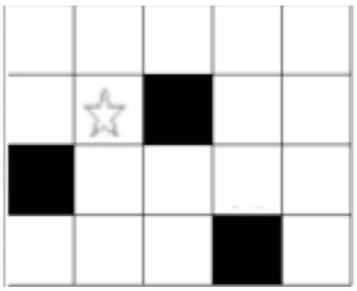
def rc1(s):
  assert(len(s) == 5)
  t = chr(ord("a")+ (ord(s[0])- ord(s[-1])))
  assert(t == "d")
  def f(s, t):
  if(s.endswith(t)):
  return [ ]
  else:
  if(ord(s[0]) > ord(t)):
  return [s[0]] + f(s[1:], t)
  else:
  return [ ] + f(s[:-1], t)
  return f(s, t) == ["f", "g", "h"]
	s =_____

3. Short Answer (5 points)

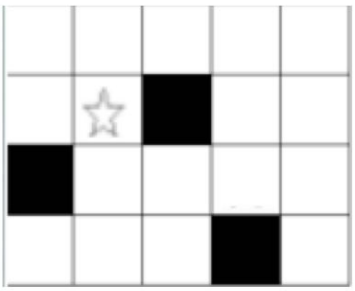
A. Fill in the blank for the solution of Towers of Hanoi:

```
def hanoi(n, source, target, temp):
    if (n == 1):
        print((source, target), end="")
    else:
        move(_____)
        move(_____)
        move(_____)
```

B. Say we start with a 4x5 board and fill the black squares as shown below (in the code from our notes, the black squares would be green, and the white squares would be cyan). Then we right-click where the star is to start a floodFill from there. On the left, write the numeric labels for the depths that result in each cell after the floodFill completes. On the right, write the numeric labels for the ordinals.



depths



ordinals

Hint #1: the numbers on the left are depths, so some labels may occur more than once.
Hint #2: the numbers on the right are ordinals, so labels must occur only once each.
Hint #3: the first label is 0, not 1 (so a 0 should be where the star is).
Hint #4: Our floodFill code recursively tries to go up, then down, then left, then right.

C. What did we do to speed up our recursive Fibonacci function and why did it make the function run so much faster?

D. What does a hash function do?

E. Give an example of MVC violation

F. What is the main difference between (type(x) == A) and isinstance(x, A)?

4. Free Response: stateList(d) [15 points]

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.

This page is left blank intentionally for your stateList(d) function

5. Free Response: parenContent(s) [15 points]

Given a string that contains a single pair of parenthesis, compute recursively a new string made of only of the parenthesis and their contents, so "xyz(abc)123" yields "(abc)". Do not worry about nested parenthesis (as there is only a single pair of parenthesis), or there is unmatched parenthesis. No iterations (for or while loop) allowed. Here are some more test cases for you:

```
assert(parenContent("15(rudina)112") == "(rudina)")
assert(parenContent("nice(memes)") == "(memes)")
assert(parenContent("(hi)") == "(hi)")
```

6. Free Response: TA, and Kosbie classes [20 points]

```
t1 = TA("Andrew", 5, "teal")
assert(t1.name == "Andrew")
assert(t1.number == 5)
assert(t1.favColor == "teal")
assert(t1.pastNames == set())  # all previous names of this TA are
contained in a set

# Two TAs are the same if and only if they have the same course
number
t2 = TA("Corey", 17, "blue")
assert(t1 != t2)
t3 = TA("Andrew", 1, "teal")
assert(t1 != t3)
t4 = TA("Nikolai", 5, "magenta")
assert(t1 == t4)
assert(str(t1) == "Andrew is TA number 5 with an uncanny love for
teal")

s = set()
assert(t1 not in s)
s.add(t1)
assert(t1 in s)
assert(t2 not in s)
assert(t4 in s)  # think about how we defined equality

t1.changeName("Reginald")
assert(t1.name == "Reginald")
assert(t1.pastNames == set(["Andrew"]))
assert(t1 in s) # still a TA even though names are changed!

koz = Kosbie()
assert(not isinstance(koz, TA))
assert(koz.name == "David")
assert(Kosbie.TAs == []) # note the it's Kosbie.TAs, not koz.TAs
Kosbie.hire(t1)
Kosbie.hire(t2)
assert(Kosbie.TAs == [t1, t2])
Kosbie.hire(t4)
assert(Kosbie.TAs == [t1, t2]) # you can't hire a TA if they have the
same number as an existing TA
Kosbie.fire(t2)
assert(Kosbie.TAs == [t1])
```

This page is left blank intentionally for your TA and Kosbie Classes

This page is left blank intentionally for your TA and Kosbie Classes

7. Free Response: playCircleGame [23 points]

Assuming the run() function is already written for you, write init, keyPressed, mousePressed, redrawAll, and timerFired so that when the animation is first run:

A. A large square is centered in the canvas
B. A score of zero is displayed in the upper right hand corner
C. A small circle, whose radius is one-fourth the size of a side of a square, appears in the left-middle of the canvas.

Gameplay proceeds as such:

A. The square starts moving down, and when it reaches the end of the window, it reappears at the top
B. The circle starts sweeping horizontally, changing directions every time it hits the end of the window.
C. If you click inside the circle, the score goes up by five, and the circle starts moving faster.
D. You can move the circle vertically using the “Up” and “Down” arrow keys
E. Whenever the circle is completely enclosed within the square, the score decreased by one, and the circle is set back to its original position.
F. If the user ever gets to 25 points, gameplay stops, and a game over message will display in the center of the screen.

Make reasonable assumptions for anything not specified here, and in any case avoid hardcoding values (such as data.width, data.height, or data.timerDelay).

This page is left blank intentionally for your playCircleGame functions()

This page is left blank intentionally for your playCircleGame functions