15-112 Midterm II

Practice Problems

Name:	____________________
Section:	____________________
andrewID:	____________________

- This PRACTICE midterm is not meant to be a perfect representation of the upcoming midterm!
- You are responsible for knowing all material covered in homework and lectures.
- As on the exam, read instructions carefully, and show all work. We will not entertain any question, except for questions pertaining to English vocabulary.
- Short answer questions should be answered briefly, but completely.
- Show your work.
- For solutions, attend a CA-led review session, or contact your CA.
- Good luck!

Time: 80 minutes

Question	**Max**	**Grade**
1	25	
2	15	
3	10	
4	10	
5	20	
6	20	
Bonus	5	
Total	105	

1. **[25 pts; 5 pts each] Indicate what each will print or draw (assume a 400x400 canvas):**
   For graphics output, do not explain the output in words, but simply draw the resulting canvas.

```
class A(object):
    def __init__(self, x): self.x = x
    def __str__(self): return "A%d" % self.x
class B(A):
    def __init__(self):
        super(B, self).__init__(5)
    def __str__(self): return "B%d" % self.x
class C(B):
    def __init__(self, a): self.x = 2*a.x
def p(a):
    # Note: each call to p(a) prints 1 line
    print a,
    print type(a).__name__,
    print 1*isinstance(a,A),
    print 1*isinstance(a,B),
    print 1*isinstance(a,C)
p(A(1))
p(B())
p(C(A(3)))
```

```
canvas.create_rectangle(50, 50, 350, 350)
canvas.create_line(50, 50, 350, 350)
canvas.create_oval(100, 200, 300, 400,
                   fill="gray")
k = 0
location = N
for c in "ABCDEF":
    location = S if location == N else N
    canvas.create_text(350-k, 50, text=c,
                       anchor=location)
    k += 60
```

```
def f(x, y):
    print "f", x, y  # don't miss this!
    if (x*y < 10):
        return [(x,y)]
    else:
        return f(x/3, y) + f(x/2, y/2)
print f(6,5)
```

```
def f(A):
    if (len(A) < 2): return A
    return [A[A[-1]%len(A)]] + f(A[:-1])
print f([2,3,5,7,11])
```

```
print [[1,[2,3],4],5][0:3][1]
print [1,{1:2},{1:3}][1:3][1][1]
print [1*3,'1'*3,{1:3},[1]*3][3][1]
print {1:['x','y'],7:'hey',9:1}[1][1]+'ou'
print {'hey':'won','I':'got','do':'it'}\
      [{'I':'hey','you':'do'}['I']]
```

2. [15 pts; 3 pts each] Very Short Answers:

a. Consider the following failed attempt to memoize fib:

```
def fib(n): return 1 if (n < 2) else fib(n-1)+fib(n-2)
memoizedValues = { }
def memoizedFib(n):
    if (n not in memoizedValues):
        memoizedValues[n] = fib(n)
    return memoizedValues[n]
print memoizedFib(40)
```

When this code runs, it works in that it prints the correct value, but it takes a really long time, so the memoization part really did not work. Very briefly, why not?

b. Here is the powerset code from the class notes, with parts of two lines removed:

```
def powerset(a):
    # returns a list of all subsets of the list a
    if (len(a) == 0):
        return [[]]
    else:
        allSubsets = [ ]
        for subset in ____________________________________:
            allSubsets += [subset]
            allSubsets += ______________________________________________
        return allSubsets
```

Fill in the blanks with the missing code.

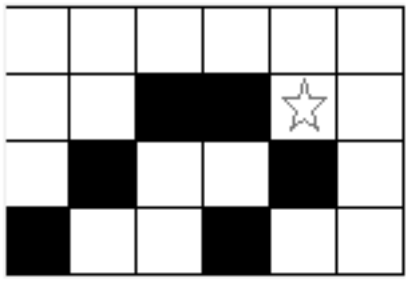
c. In our Tetris design, rather than first test if a rotation is valid, we instead just do the rotation, then *undo* it if it was not valid. In just a few words, why did we choose this approach?

d. Here is a sort code from the class notes (with the function renamed "f"):

```
def f(L):
    if (len(L) < 2):
        return L
    else:
        first = L[0]
        rest = f(L[1:])
        lo = [x for x in rest if x < first]
        hi = [x for x in rest if x >= first]
        return  lo  +  [first]  +  hi
```

Make just a couple very simple edits to this code so that it implements quicksort.

e. Say we start with a 4x6 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. Write the numeric labels that result in each cell after the floodFill completes. Note that the floodFill code is slightly modified with the changes shown below:



```
def floodFill(row, col, color, depth=0):
        …
        floodFill(row-1, col-1, color, depth+1)
        floodFill(row-1, col+1, color, depth+1)
        floodFill(row+1, col+1, color, depth+1)
        floodFill(row+1, col-1, color, depth+1)
```

3. **[10 pts; 3 pts each, 1 free point] Reasoning about code**

In these exercises, you should provide the arguments that cause the function to return the specified value. In most cases, there are many valid answers. You only need to provide one of them.

```
def f(a, b):
    z = 0
    False = (bool)(0)
    for x in xrange(a**b):
        if (x % b == 0):
            True = False
            False = not True
        else:
            z += a
    return (False and z > a**b)

# provide a value of a and b such that f(a,b) returns True
```

```
def f(n):
    if (n == 0):
        return (0,0)
    else:
        (x,y) = f(n/10)
        return (x+1,y+(n%10))

# provide a value of n where f(n) returns (5,43)
```

```
def f(x):
    if(len(x)<=2):
        return 0
    elif(x[0] <= 1):
        return 10
    else:
        return len(x)-f(x[::x[0]])+f(x[::2])

# provide a value of x such that f(x) returns 9
```

4. **[10 pts] Free Response: recursion 1**
   Write the function listEmptyFolders(path) that returns an alphabetized list of the names (*not the paths*) of all the empty folders starting from the given path, where an empty folder does not contain any files or other folders. If the path is not a folder (or not even a string), listEmptyFolders(path) should not crash, but should just return the empty list.

5. **[20 pts] Free Response: recursion 2**

Write the function subsetSum(a,n), which takes in a list a of integers (might be negative!), and a number n, and returns true if and only if there's a sublist of the numbers in a that sums to n. For example:

subsetSum([1,2],1) = subsetSum([1,2],3) = subsetSum([1,2],0) = True (the empty list is a sublist!) subsetSum([1,2],-1) = subsetSum([1,2],4) = False.

6. **[20 pts] Free Response: animation**

Assuming the canvas is 500x500, write run, init, timerFired, keyPressed, and redrawAll to make a simple analog stopwatch with a second hand (but no minute hand or hour hand). Your redrawAll should create only two objects – a circle and a line. At the start, and whenever the user presses 'r' (for 'reset'), the second hand points directly up and does not move. Then, when the user presses 'g' (for 'go'), the hand starts moving clockwise (of course) at the rate of one revolution per minute. When the user presses 's' (for 'stop'), the hand stops but does not reset to 0. Subsequent presses of 'g' and 's' will make the hand go and stop again, all without resetting. Your solution must move smoothly, even though calls to timerFired may vary in frequency, so you must use time.time(), which returns the number of seconds since some arbitrary time many years ago. Do NOT use global canvas.

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

```
a = [1,2]
def f(x,y):
    if (x == f): return y
    elif (type(x) == type(f)): return x(f(y,y),y)
    try: return y(a,a[0])
    except: return 2+f(f,x)
def g(x,y): return 1+x*y
print f(g,f(g,1))
```

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

```
def f(g): return lambda x: g(*range(x,x+3))
@f
def g(*f): return (f[0]+f[1],f)
print g(4)
```