

Name: \_\_\_\_\_ Recitation: \_\_\_\_\_ Andrew Id: \_\_\_\_\_

**15-112 Spring 2019 Quiz 9**

Up to 20 minutes. No calculators, no notes, no books, no computers. Show your work!

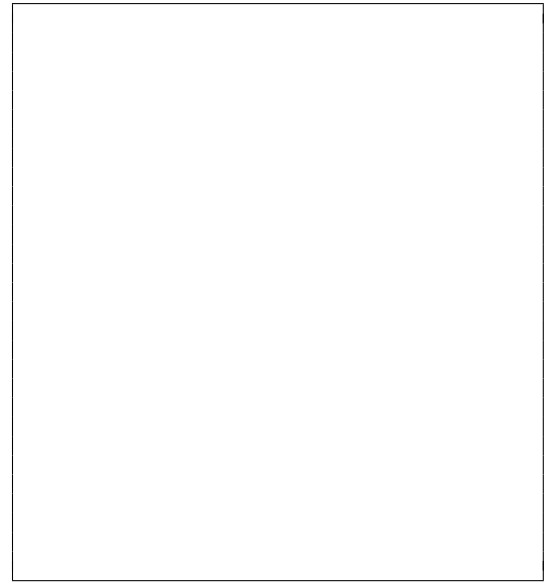
1. (20 points) **Free Response:** Write the function `powerSum(n, k)` that takes two positive integers, `n` and `k`, and returns the result of the following power-sum sequence:  $1^k + 2^k + \dots + n^k$ . For example, `powerSum(4, 2)` would return `30`, because  $1^2 + 2^2 + 3^2 + 4^2 = 1 + 4 + 9 + 16 = 30$ .

**Your solution must use recursion meaningfully. Any solution which uses loops, generators, or comprehensions will automatically receive a 0.**

2. (30 points) **Code Tracing:** Indicate what the following program prints. Place your answer (and nothing else) in the box to the right of the code.

```
def ct(x, y, depth=0):
    print("%d in: %d, %d" % (depth, x, y))
    if x < 10:
        result = x
    elif y < 10:
        result = y
    elif abs(x - y) < 10:
        tmp = ct(x // 10, y // 10, depth + 1)
        result = (x % 10) + (y % 10) + tmp
    else:
        a = (x % 10) + ct(x // 10, y, depth + 1)
        b = (y % 10) + ct(x, y // 10, depth + 1)
        result = a + b
    print("%d out: %d" % (depth, result))
    return result

ct(112, 15)
```



3. (30 points) **Free Response:** Write the function `flattenStrings(lst)`, which takes a list which may contain lists (which themselves may contain lists, and so on) and returns a one-dimensional list containing each of the strings, in the original order, that appeared in some place in the original list. For example, `flattenStrings([ 'a', [ 'b' ], [ 'c', [ 'd', 'e' ]], 'f' ], 'g' ])` would return `[ 'a', 'b', 'c', 'd', 'e', 'f', 'g' ]`.

Note that the original list may contain some values which are not lists or strings; these should be ignored. For example, `flattenStrings([ 'wow', [ 2, [ [ ] ] ], [ True, 'gosh' ] ])` should return `[ 'wow', 'gosh' ]`.

**Note:** it's okay to use loops in this problem, though you should still use recursion in some way.

4. (20 points) **Reasoning Over Code:** Find an argument for the program `rocWrapper` that makes it return `True`. Place your answer (and nothing else) in the box to the right of the code.

```
def roc(lst):
    if len(lst) == 0:
        return [ ]
    elif type(lst[0]) == int:
        return [ lst[-1] ] + roc(lst[1:-1])
    else:
        return roc(lst[1:])

def rocWrapper(lst):
    assert(len(lst) % 2 == 1 and len(lst) <= 8)
    return roc(lst) == [ 1, 2, 3 ]
```

