Name: _____ Date: _____

Pledge: _____

Total: ____ / 75 = ____

Closed book: no textbook, no electronic devices, one sheet of paper with notes. Read each question carefully before answering, as there is **no partial credit for any response**! You may work out the solution to each question within the test itself, but **only your answers on this cover page will be graded**.

Question 1

['80', 'd', 'f'] (5 points)

Question 2

up (5 points)

Question 3

8 (5 points)

Question 4

a) **14** (5 points)
b) **(d) tree** (5 points)
c) **15** (5 points)
d) **return 2 * hanoi(n-1) + 1** (5 points)
e) **(c) linear** (5 points)

Question 5

a) **6** (5 points)
b) **(c) linear** (5 points)
c) **balon** (5 points)

Question 6

a) **[]** (3 points)
b) **collapse(lst[0])** (3 points)
c) **collapse(lst[1:])** (3 points)
d) **[lst[0]]** (3 points)
e) **collapse(lst[1:])** (3 points)

Question 7

lambda x: x > threshold, map(lambda x: x * x, lst) (5 points)
x ** 2 also accepted

Question 8

(a) tail (5 points)

Question 1 (5 points)
Consider the following code:

```
L = ['a', 'b', 'c', 'd', 'f']
M = ['90', '80', '70', '60', '0']
N = [ M[1] ] + L[3:]
```

What is the value of N after these statements have executed?

Question 2 (5 points)
Consider the following code:

```
L = ['jack', 'and', 'jill', 'went', 'up', 'the', 'hill']
M = range( 2, len(L), 2 )
print(L[ M[ 1 ] ])
```

What is printed on the screen after these statements have executed?

Question 3 (5 points)
Consider the following code:

```
L = [1, 3, 5]
M = L + [2, 4, 6]
N = map(lambda x: x ** 3, M)
```

What is the value of N[3] after these statements have executed?

Question 4 (25 points)
Consider the call to `hanoi(n)` for the function definition below.

```
def hanoi(n):
    if n == 1:
        return 1
    return hanoi(n – 1) + 1 + hanoi(n – 1)
```

a) Excluding the call to `hanoi(4)`, how many recursive calls are made before this function terminates?

b) What type of recursion is found? Select the **best, most specific** answer.
   (a) tail
   (b) mutual
   (c) linear
   (d) tree
   (e) nested

c) What is the final value returned by `hanoi(4)` ?

d) Rewrite the last line of the function
```
        return hanoi(n – 1) + 1 + hanoi(n – 1)
```
   to be more efficient.

e) Using your answer in part d), what type of recursion is found in the function now? Select the **best, most specific** answer.
   (a) tail
   (b) mutual
   (c) linear
   (d) tree
   (e) nested

Question 5 (15 points)
Consider the following code:

```
def mystery(s):
    if len(s) <= 1:
        return s
    if s[0] in s[1:]:
        return mystery(s[1:])
    return s[0] + mystery(s[1:])

print(mystery('balloon'))
```

a) Excluding the call to `mystery('balloon')`, how many recursive calls are made before this function terminates?

b) What type of recursion is found? Select the **best, most specific** answer.
   (a) tail
   (b) mutual
   (c) linear
   (d) tree
   (e) nested

c) What does `mystery('balloon')`return?

Question 6 (15 points)
Implement the following function using recursion:

```
def collapse(lst):
    '''Assume lst is a list of values, some of which may also be lists.
    Returns a list with all the values collapsed. In other words, nesting has
    been removed and the values appear in the same order as the original
    list, from left to right. You may not use negative indexing or slicing.
    Examples:
    collapse([]) -> []
    collapse([1, [2, 3], [[4, 5], [6]], 7]) -> [1, 2, 3, 4, 5, 6, 7]
    '''

    if lst == []:
          return ______(a)______
    if isinstance(lst[0], list):
        return _____(b)_____ + _____(c)_____
    return ______(d)______ + ______(e)______
```

Question 7 (5 points)
Implement the following function using **map**, **filter,** and **lambda**.

```
def keep_large_squares(lst, threshold):
    '''Assume lst is a list of integers.
    Returns a list of integers that are the squares of the original values
    exceeding the supplied threshold.
    Examples:
    keep_large_squares([], 10) -> []
    keep_large_squares([-10, 2, 3, 20], 50) -> [100, 400]
    '''

    return filter(_____________________________________________________)
```

Question 8 (5 points)

```
def confuse(n):
    if n % 20 == 0:
        print('Yay!')
        return
    print('working')
    confuse(n - 1)
```

What type of recursion is found? Select the **best, most specific** answer.

(a) tail
(b) mutual
(c) linear
(d) tree
(e) nested