Name: _____ Date: _____

Pledge: _____

Closed book: no textbook, no electronic devices, one sheet of paper with notes. Read each question carefully before answering! Write your answers on the test paper and turn in your notes.

Question 1 (5 points)
Consider the following code:

```
A = [ 3, 1, 2 ]
L = ['This', 'is', 'a', 'fun', 'list']
M = ['not', 'possibly', 'definitely']
N = L[ :-3 ] + [ M[2] ] + [ M[0] ] + [ L[ A[0] ] ]
```

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

```
['This', 'is', 'definitely', 'not', 'fun']
```

Assess: [state]
Rubric: (1 point for list brackets, 1 point for strings in quotes, 2 points for correct values, 1 point for no additional values)

Question 2 (5 points)
Consider the following code:

```
L = ['www', 'stevens', 'edu', 'sit', 'registrar']
M = range( len(L), -1, -2 )
print(L[ M[ 2 ] ])
```

What is printed on the screen after these statements have executed?
```
stevens or 'stevens'
```

Assess: [state]
Rubric: (3 points for sit or 'stevens', 2 points for no additional values)

Question 3 (5 points)
Consider the following code:

```
L = ['www', 'stevens', 'edu']
M = ['www', 'brown', 'edu'] + L
N = filter(lambda s: len(s) > 3, M)
```

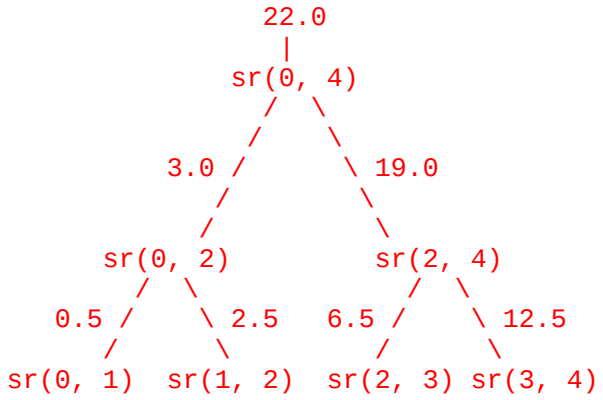
What is the value of N after these statements have executed?
```
['brown', 'stevens']
```

Assess: none
Rubric: (1 point for list brackets, 1 point for string values, 2 points for correct string values, 1 point for no additional values)

Question 4 (20 points)
Consider the following function:

```
def simpson_rule(a, b):
    if b - a <= 1:
        return (a * a + b * b) / 2
    m = (a + b) / 2
    return simpson_rule(a, m) + simpson_rule(m, b)
```

a) Trace `simpson_rule(0, 4)` in the space below. Be sure to show each recursive call, the arguments to the function, and the returned values. (10 points)



b) What value does `simpson_rule(0, 4)` return? (5 points)

- ❍ A. 4.0
- ❍ B. 6.0
- ❍ C. 20.0
- ❍ D.✓ 22.0
- ❍ E. 30.0

c) Function `simpson_rule` is an example of which type of recursion? (2 points)

- ❍ A.✓ tree
- ❍ B. linear
- ❍ C. nested
- ❍ D. tail

d) How many recursive calls are made when calling `simpson_rule(0, 4)`. Do not count the initial call to `simpson_rule(0, 4)` in your answer. (3 points)

Answer: 6

Assess: [execution]
Rubric:

a) (2 points for a tree with 2 branches per node, 3 points for some part of the tree drawn correctly, 5 points for correct tree with all return values shown; 10 points total)
b) (5 points for d, all or nothing)
c) (2 points for a, all or nothing)
d) (3 points for 6, all or nothing)

Question 5 (20 points)
Consider the following code:

```
def process(s):
    if len(s) > 1:
        s = process(s[1:]) + s[0]
    return s
```

a) Trace `process('spam')`in the space below. That is, starting with `process('spam')`, show each call that is made, with the argument values, in a downward growing stack. In addition, draw a reverse arrow labeled with the value returned, for each call. (15 points)

```
process('spam')
   ➔ process('pam') + 's' = 'maps'
process('pam')
   ➔ process('am') + 'p' = 'map'
process('am')
   ➔ process('m') + 'a' = 'ma'
process('m')
   ➔ 'm'
```

Assess: [execution]
Rubric: (3 points for first function call, 3 points for returning 'm' in base case, 5 points for going up stack, 5 points for correct result)

b) What does the process function do? (5 points)

- ❍ A. Places the first character of s at the end
- ❍ B. Places the last character of s at the beginning
- ❍ C.✓ Reverses s
- ❍ D. Returns an exact copy of s
- ❍ E. Goes into infinite recursion and overflows the stack

Assess: [execution]
Rubric: (5 points, all or nothing)

Question 6 (20 points)

a) Complete the following function using recursion:

```
def triangular(n):
    '''Assume num is a positive integer.
    Returns the nth triangular number. The sequence of triangular
    numbers is 1, 3, 6, 10, 15,... Pay special attention to how you can
    use the previous number in the sequence to generate the next one.
    Examples:
    triangular(1) -> 1
    triangular(2) -> 3
    triangular(4) -> 10
    '''
    if n == 1:
        return 1
    return n + triangular(n – 1) (10 points)
```

Assess: [coding]
Rubric: (
4 points for calling triangular function,
3 points for argument n - 1,
3 points for adding n
)

b) Implement the following function using **map** and **lambda**. Without map and lambda, no credit will be given. You may **NOT** use the function `triangular(n)` that you wrote in the first part of the question.

Hint: A shortcut to the nth triangular number is found by applying the formula: $\frac{n(n+1)}{2}$

```
def triangular_numbers(lst):
    '''Assume lst is a list of positive integers.
    Returns a list of 2-element lists (or pairs) where the first number n
    is from lst and the second is the nth triangular number.
    Examples:
    triangular_numbers([]) -> []
    triangular_numbers([3]) -> [[3, 6]]
    triangular_numbers([1, 2, 3, 4]) -> [[1, 1], [2, 3], [3, 6], [4, 10]]
    '''

    return map(lambda n: [n, n * (n + 1) // 2], lst) (10 points)
```

Assess: [coding]
Rubric: (
2 points for correct use of map,
1 point for lambda n,
2 points for returning 2-element list from lambda,
1 point for n as first element,
3 points for correct formula [1 point for //] as second element,
1 point for passing lst
)

'''

```
CS 115 A, Fall 2016 - Test 1, Questions 7 and 8

Author: <your name here>
Pledge: <write pledge>
'''
from cs115 import filter
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' RULES: You can use Canvas to download this file and upload your solution.
' You can use Eclipse to edit and run your program.  You should NOT look at
' other programs in Eclipse, you should NOT use any other programs, and you
' should NOT use any notes or books.
' According to the Honor Code, you should report any student who appears
' to be violating these rules.
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
' Question 7 (20 points)
' Implement these functions using recursion.
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
def reverse(s):
    '''Reverses a string.
    This part is worth 5 points.'''
    if s == '':
        return ''
    return reverse(s[1:]) + s[0]
```

Rubric:
(2 points for correct handing of base case,
2 points for calling reverse with correct argument,
1 point for adding first character to the end of the recursive call and returning
)

```
def keep_palindromes(lst):
    '''Assume lst is a list of strings where all letters are lower case.
    Return a list of strings that are palindromes. A palindrome is a word
    that has the same spelling both forward and backward. For instance,
    'racecar' is a palindrome. You may use the reverse function written
    above.
    This part is worth 15 points.'''
    if lst == []:
        return []
    if lst[0] == reverse(lst[0]):
        return [lst[0]] + keep_palindromes(lst[1:])
    return keep_palindromes(lst[1:])
```

Rubric:
(3 points for correct handling of base case,
2 points for comparing word and its reverse for equality,
6 points for correct recursive calls (3 points each),
3 points for concatenating a palindrome to the list,
1 point for return statements
)

```
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
```

```
' Question 8 (10 points)
' Implement this function using the Python's built-in 'filter' and 'lambda'.
' DO NOT USE recursion.
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
def keep_palindromes_filter(lst):
    '''Assume lst is a list of strings where all letters are lower case.
    Return a list of strings that are palindromes. A palindrome is a word
    that has the same spelling both forward and backward. For instance,
    'racecar' is a palindrome. You may use the reverse function written
    above.'''
    return filter(lambda s: s == reverse(s), lst)
```

Rubric:
(5 points for correct use of filter (all or nothing),
5 points for correct lambda function
)