Answer the questions in the spaces provided. If you run out of room for an answer, continue on the back of the page.

Question	Points	Score
1	3	
2	6	
3	15	
4	10	
5	15	
6	20	
Total:	69	

Name: _____

Section: _____

1. **Survey**

   (a) (1 point) What was your favorite part of the class?

   (b) (1 point) What was your least favorite part of the class?

   (c) (1 point) What would you do to improve the class for future semesters?

2. **Short answer**

   (a) (3 points) Describe list comprehension, in terms of functional tools. For what tool is list comprehension syntactical sugar?

   (b) (3 points) Describe the way that `reduce` works. What kind of function does it take as a parameter?

3. **One-liners**

   (a) (3 points) Using `reduce`, write a one-line function to concatenate a nested list of depth 1 into a single list.

   (b) (6 points) Write one line of code that prints out a list of the first 100 *odd* squares.

   (c) (6 points) Recall that the cumulative sums of a list L is another list $L_{\text{cumulative}}$ of length $|L|$, defined index-wise as follows:

$$L_{\text{cumulative}}[i] = \sum_{j=0}^{i} L[j]$$

   Let the list from part (a) be denoted L_a. Write one line of code that prints out the cumulative sums of the list from part (a).

4. (10 points) Recall from the previous practice exam the Fibonacci numbers, defined recursively as follows:

$$F_n = \begin{cases} F_{n-1} + F_{n-2} & n \geq 2 \\ 1 & n = 1 \\ 0 & n = 0 \end{cases}$$

Using only **recursion**, write an ***efficient*** function to calculate the n^{th} Fibonacci number. The function header (not necessarily the first line of code) is written for you.

```
def fib(n):
```

5. (15 points) Write code for a generator that yields only the squares of the Fibonacci numbers divisible by 3.

6. **Binary search** is a method of finding a certain element in a sorted list L. In binary search, you repeatedly divide the list in two and then determine which side the target element lies on. After partitioning L in half for $\log(|L|)$ times, the target element is found.

   Consider the following list:

   $$L = [1, 2, 4, 7, 10, 15, 63, 72, 81, 94, 97]$$

   Say we have target element 63. First, we find the center element of 15. Since $63 > 15$, we continue on the right side:

   $$[63, 72, 81, 94, 97]$$

   The next middle element is 81. Since $63 < 81$, we look to the left and find $[63, 72]$. Then, since we're down to two elements, we can trivially select 63 after some quick comparisons. (This logic is left to you to figure out on your own. Note that no special logic would be required if there were 3 elements instead of 2 – the parity of $|L_i|$ in iteration i is important in your logic.)

   (a) (10 points) Implement binary search **iteratively**.

   (b) (10 points) Implement binary search **recursively**.