Sequences Worksheet II
CS51 - Fall 2026

Practice Problem 1 - Sorting strings

You are given the following function definition:

```
def compare_class(class_number):
    if class_number < 'cs51':
        print(class_number, 'comes before cs51.')
    elif class_number > 'cs51':
        print(class_number, 'comes after s51.')
    else:
        print('Hello, cs51')
```

What will happen if you call `compare_class('cs50')` ?

What will happen if you call `compare_class('CS62')`?

Practice Problem 2 - String functions

Define a function `str_odd_indices` that takes one parameter `s` (a string) and returns a string comprised of only the odd indexed characters of `s`. For example, `str_odd_indices('hello!')` would return `'el!'`.

Practice Problem 3 - More string functions

Define a function `find_char` that takes two parameters, a string `s` and a character `c` and returns the index of the first instance of that character. If that character does not appear in the string, it returns -1.

For example:

```
find_char('hello', 'h') == 0
find_char('hello', 'l') == 2
find_char('hello', 'a') == -1
```

Practice Problem 4 - Recursion and strings

Write a recursive function called `reverse` that takes a string as a parameter and reverses the string.

Practice Problem 5 - Recursion and lists

Write a recursive Python function called `rec_linear_search` that takes as input a list, an element, and an index that has a default value of 0, and returns the index of the first encounter of the element in the list, if it exists, or -1 if it does not.

Practice Problem 6 - Mystery function

What does this mystery function do?

```
def rec_mystery(l):
if len(l) == 1:
   return l[0]
else:
    m = rec_mystery(l[1:])
    if m > l[0]:
        return m
    else:
        return l[0]
```

Practice Problem 7 - Loop invariants

Given a list `my_list` (of size `n`) of numbers, write an iterative function `sum_of_list_numbers` that calculates the sum of the numbers in `my_list`.

1. What are the pre-and post-conditions?

2. What is a good loop invariant?

3. Use loop invariants to prove that your function works correctly.

Practice Problem 8 - Aliasing

You are given the function

```
def augment_twice(a_list, val):
    a_list.append(val)
    a_list.append(val)
```

What happens to `numbers` after executing the following code?

```
numbers = [1,2,3]
augment_twice(numbers, 47)
```

What if we were to change the function to this and execute the same code?

```
def augment_twice(a_list, val):
    a_list = a_list + [val, val]
```