6.189 Worksheet
Session 9

Administrivia

Name:

Instructions:

1. Err..complete the questions :).
2. No calculators, no laptops, etc.
3. When we ask for output, you DON'T have to write the spaces/newlines in.

Program Text:

```
print “X”,
print “X”,
```

Output:

```
XX
```

Problem 1: Common Errors

Each of the following code snippets represents a common error made by an introductory programming student (I know this because I ripped these errors from an introductory programming website.) What did they do wrong?

Assume that the following definitions occur before every code snippet.

Program Text:

```
my_string = "This is a sentence."
my_list = [4,2,6,8]
user_input = "100"
my_integer = 27
```

1. Convert `my_string` to lowercase

Program Text:

```
my_string.lower()
```

Answer:

2. Print every element in `my_list` in *reverse* order.

Program Text:

```
for i in my_list.reverse():
  print i,
```

Answer:

3. Reverse the order of elements in `my_list`.

Program Text:

```
my_list.reverse
```

Answer:

4. `user_input` contains a string representation of a number entered by the user. Multiply it by 10 and add the resulting value to `my_list` as a string.

Program Text:

```
bigger_input = user_input + 0
my_list.append(user_input)
```

Answer:

5. Create a backup copy of `my_list`, then remove the largest element from `my_list`.

Program Text:

```
new_list = my_list
my_list.remove(max(my_list))
```

Answer:

5. This function finds the position of the *last* instance of an element `e` **in a list. Hints: 1) There are no syntax errors 2) The function always returns the correct value.**

Program Text:

```
def rindex(my_list, e):
  """Finds the position of the last occurrence of e in my_list. If e
is not in l, returns -1"""
  if e not in my_list:
    return -1
  my_list.reverse()
  for i in range(len(my_list)):
    if my_list[i] == e:
      return len(my_list) - 1 - i
```

Answer:

6. Prints all elements of `my_list`

Program Text:

```
for i in my_ list:
  print i
  i = i + 1
```

Answer:

7. Finds the largest element in a list

Program Text:

```
def find_max(list):
  """Finds the largest integer in list. Assumes list contains only
positive integers"""
  max = 0
  for i in my_list:
    if i > max:
      return i
  return max
```

Answer:

Problem 2: Meaningful Names!

Disclaimer: This example is a bit exaggerated.
You'll learn more about programming style in subsequent courses, but one thing we want to imprint in you now is using *meaningful* variable names. Every variable is created for a reason – its name should reflect the values you choose to store in it.

The following code is something an introductory student could have written for a class. Imagine being the TA trying to find the bug in it.

Program Text:

```
def f3(ll):
  #all stuff in ll between 0 and 1000000
  j = 0
  k = 0
  for i in range(len(ll)):
    if ll[i] > ll[j]:
      j = i
    elif ll[i] < ll[k]:
      k = i
  l = ll[j]
  ll[k] = ll[l]
  ll[j] = ll[k]
```

Your task is to find the bug in the above code. The function should swap the maximum and minimum elements in a list.

Well..maybe that's too mean. Here – I'll give you some meaningful variable names for the above code.

$f3 \rightarrow$ swap_max_min	$j \rightarrow$ max_position	$l \rightarrow$ temp
$ll \rightarrow$ list	$k \rightarrow$ min_position	

Answer:

Problem 3: Test Cases

We've written a function that calculates the square root of a number. If given a negative number, our function returns 0 (if you ever write a square root function, don't do that :p.) We're using this function in a much larger program that controls our 6.01 robot, so its kind of important that the know the function works correctly.

How can we tell if a function works correctly? Staring at it for 30 minutes is probably not the best solution..Instead, we're going to write a couple of test cases to make sure it works.

Program Text:

```
SQ_3 = ... #assume SQRT_3 has been initialized to the square root
           #of 3 (1.717...)


test_cases = [_____, _____, _____, _____, _____, _____]

test_case_answers = [_____, _____, _____, _____, _____, _____]

def custom_sqrt(num):
  "Returns the square root of num, or 0 if num < 0"
  ... (code snipped)

for i in range(len(test_cases)):
  if custom_sqrt(test_cases[i]) != test_case_answers[i]:
    print "Test Case #",i,"failed!"
```

Note: You can use the built-in `zip` function to make the last three lines prettier ... look it up at some point.

Fill in the blanks below to complete our testing code. You'll want to use as many unique cases as possible – don't just test 1,2,3,4,5,6. At least one of your test cases should be a negative number, for example.

We gave you six blanks, but you don't need to use all of them. You want a wide variety of test cases in order to catch as many bugs as possible, but you also don't want to test every random number you can think of. It's a delicate balance :) – just do your best.