CS 115 Exam 3, Spring 2012

Your name: _____

Rules

- You may use one handwritten 8.5 x 11" cheat sheet (front and back). This is the only resource you may consult during this exam.
- Explain/show work if you want to receive partial credit for wrong answers.
- As long as your code is correct, you will get full credit. No points for style.
- When you write code, be sure that you clearly indicate the indentation level of each statement.

Grade (instructor use only)

	Your Score	**Max Score**
Problem 1		25
Problem 2		15
Problem 3		30
Problem 4		30
Total		100

Problem 1: 25 points.

What will print to the screen when each of the following snippets of code is executed in IDLE?

Be very clear with spacing, line breaks, etc.

Note: the parts of this problem are *independent*.

For all parts of this problem, assume that the following functions have been defined.

```
def f1( ):
    return 12

def f2(x, y):
    return 2 * x + y

def f3(z):
    return f2(z, z+1)

def f4(x):
    x = 4
    return x ** 2
```

(a)

```
print(f1())
```

(b)

```
print(f2(3, 1))
```

(c)

```
print(f3(1))
```

(d)

```
a = 5
print(f4(a))
print(a)
```

(e)

```
x = f1()
print(x)
```

Problem 2: 15 points.

Consider the following sorted list:

```
cities = ['Cairo',
          'Lima',
          'London',
          'New York',
          'Paris',
          'Rome',
          'Seoul',
          'Sydney',
          'Tokyo']
```

and the following binary search (which is essentially identical to your lab code):

```
# binary_search()
# Finds the position of an item in a list
# Parameters: the list; the item to search for
# Returns: the item's position (or None)
def binary_search(search_list, value_to_find):
     first = 0
     last = len(search_list) - 1

     while first <= last:
          middle = (first + last) // 2
          if value_to_find == search_list[middle]:
               return middle
          elif value_to_find < search_list[middle]:
               last = middle - 1
          else:
               first = middle + 1
     return None
```

You may want to label the elements of `cities` with their numeric index values before proceeding.

Answer the questions on the next page.

(a) Fill out the following table tracing a binary search for *Lima* in this list. You should fill out one row per iteration of the loop. If there are more rows than iterations, leave the extra rows blank.

In the *Compare To:* column, you should give the VALUE (the name of the breakfast item) of the list element that will be compared to *Lima*.

Old value of `first`	**Old value of `last`**	**Compare to (e.g. *Tokyo)*:**
0	8	

(b) Fill out the following table tracing a binary search for *Shanghai* in this list.

Old value of `first`	**Old value of `last`**	**Compare to (e.g. *Tokyo)*:**
0	8	

Problem 3: 30 points.

Write functions to perform the following tasks.

Keep in mind the following:

- Your functions should NOT ask the user for input.
- Your functions should NOT print anything.
- Your functions should NOT call sys.exit() to terminate the program.

(a) Write a function called `cube` that…
* has one input parameter: a number
* returns the cube of the number

(b) Write a function called `count_chickens` that…
* has one input parameter: a list of words
* returns the number of elements of the list that are equal to `chicken`

(c) Write a function called `truncate` that…

* has two input parameters: a list of words *L*, and a number *N*
- returns a new list. Each element of the new list consists of the first *N* characters of the corresponding element of *L.* If the element of *L* has fewer than *N* characters, then it is copied to the new list in its entirety.

Problem 4: 30 points.

For this problem, you must write a **complete program**. That includes a docstring, a `def main()`, any necessary library imports, etc.

Read the instructions carefully before you start coding!

Your program should contain the following:

1. A function called `ReadIntFile` that does the following:
    - Takes a filename as a parameter
    - Opens that file (you can assume the file exists)
    - Reads each line of the file in as an integer (you can assume that the file contains one integer per line)
    - Returns the list of integers read from the file

2. A function called `IsSum` that does the following:
    - Has two input parameters: a list *L* and a number *N*
    - Returns True if *N* is an element of *L* and False otherwise

3. A main function that does the following:
    - Calls `ReadIntFile` to read from the file *numbers.txt*
    - Repeatedly prompts the user to enter a number:
        - Prints *Yes!* if the user's number was in *numbers.txt*
        - Stops if the user enters a non-numeric values

[blank]