# CS 115 Exam 2, Spring 2014

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 the indentation level of each statement is clear.

| | **Your Score** | **Max Score** |
|---|---|---|
| Problem 1: Tracing snippets of code | | 25 |
| Problem 2: Tracing functions | | 15 |
| Problem 3: Defining functions | | 35 |
| Problem 4: Defining and calling functions | | 25 |
| **Total** | | **100** |

## Problem 1: Tracing snippets of code (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*.

(a)

```
v = 10
while v > 0:
    print(10 * v)
    v -= 2
```

(b)

```
L = ["Rick", "Carl", "Glenn", "Daryl"]
print(len(L))
print(L[3])
print(L[3][1])
```

(c)

```
s = "2014"
t = "1"
print(s + t)
```

(d)

```
zombies = "The Walking Dead"
more_zombies = zombies.split()
print(zombies[1])
print(more_zombies[1])
```

## Problem 2: Tracing functions (15 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 3

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

def f3(x):
    return f2(2, x)

def f4(x, y):
    return f3(y) + f1()
```

(a)

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

(b)

```
print(f2(6, 10))
```

(c)

```
print(f4(6, 10))
```

## Problem 3: Defining functions (35 points)

Define functions to perform the following tasks, obeying these requirements:

- Do NOT ask the user for input unless the specification explicitly requires it.
- Do NOT print anything unless the specification explicitly requires it.
- Do NOT call sys.exit() to terminate the program.

---

(a)

Define a function called `convert_dollars` that…

- Has one parameter: a value in dollars
- Returns the equivalent value in cents

(b) Define a function called `count_zs` that…

- Has one parameter: a list of strings.
- Returns the number of 'z' or 'Z' characters appearing in any of the strings.

For example, it should return 3 for the input `["PIZZA", "is", "zesty"]`

(c) Define a function called `check_num` that…

- Has two parameters that are integers, which are not guaranteed to be in any particular order.
- Prompts the user to enter an integer. (You can assume that the user always enters an actual integer.)
- Returns the user's number *only if* it is between the values of the parameters, *inclusive.* That is, if the user's value is equal to one of the parameters, it will still count as *between*.
- Otherwise, keeps repeating these steps until the user enters a number that is between the two parameters.

(d) Define a function called `all_caps` that...

- Has one parameter: a list of strings
- Returns `True` if ***every element of the list*** is all caps, and `False` if any element contains any lowercase letters.
- Explanations and special cases:
    - Function should return `True` if the list is empty.
    - Non-alphabetic characters should be ignored. For example, `AAARGGGH!!!` should be considered all caps despite the exclamation marks.

## Problem 4: Defining and calling functions (25 points)

For this problem, you will write three functions, which should obey the rules given in Problem 3. ***To get full credit, your functions should call each other where appropriate to avoid duplicating work.***

Define a function named `increasing` that...

- Takes a list of numbers as a parameter
- Returns `True` if each element is greater than or equal to its predecessor, and `False` otherwise. It should return `True` if the list has 0 or 1 elements.
- Examples:
  Should return `True` for `[90, 90, 90, 90, 90]`
  Should return `False` for `[91, 92, 93, 85, 94]`

Define a function named `average` that...

- Takes a list of numbers as a parameter
- Returns their mean

Define a function named `print_results` that...

- Takes a list of numbers as a parameter
- **Prints** their mean
- **Prints** "Good job!" if each element is greater than or equal to its predecessor.