Name _____

CS 121B Practice Final Exam R. Brown May 9, 2014

SHOW YOUR WORK—No work may mean no credit

I pledge my honor that I have neither given nor received assistance during this exam, and that I have seen no dishonest work.

Signed _____

☐ I have intentionally not signed the pledge (*check only if appropriate*)

Answer **7 of the following 8 problems**. Clearly mark the problem you are skipping.

Note: **Some problems are worth more points than others. Choose the problems worth more points for a higher potential score.**

(*6 pts*) 1. Write a Python3 function `report()` that satisfies the following spec. Use iteration, with no recursion. **Include an invariant for your loop.**

`report`

1 Argument: A list of integers representing total cholesterol readings.

Return: A string expressing the percentages of desirable (less that 200), borderline high (between 200 and 239), and high (240 or more) cholesterol readings in the list *arg1*.

Example:

```
report([180, 250, 210, 238, 220, 245])
 --> "16.7% desirable, 50.0% borderline, 33.3% high"
```

(*8 pts*) 2. Write a Python3 function that satisfies the following spec.

`countInput`
1 Argument: A string prompt.
State change: Print the prompt *arg1*, read input lines up to the first empty line, and prints a summary of the number of words and the number of characters appearing in those lines.
Return: The number of lines read, excluding that first empty line.

- Use an empty line to indicate end of input text.
- Assume that words are separated by whitespace characters.
- *Include newline characters* in your count of characters (add one per line if needed).

Example:

```
countInput("Enter lines of input, followed by an empty line.")
 Enter lines of input, followed by an empty line.
"The time has come," the walrus said,
"to talk of many things."

 Summary:  12 words, 64 characters
--> 2
```

(*6 pts*) 3. Write a Python3 function `replaceD()` that satisfies the following spec. Use iteration *or* recursion, and include appropriate invariants and/or assert comments.

`replaceD`
2 Arguments: A list of strings and a dictionary whose keys and values are both strings.
Return: A list of strings, the same elements as in *arg1*, except with every string that appears as a key in *arg2* replaced by an upper-case version of that key's value in *arg2*.

Example:

```
replaceD(['the', 'cat', 'in', 'the', 'hat'], {'cat':'dog', 'box':'container'})
 --> ['the', 'DOG', 'in', 'the', 'hat']
replaceD(['the', 'cat', 'in', 'the', 'hat'], {'cat':'dog', 'the':'that'})
 --> ['THAT', 'DOG', 'in', 'THAT', 'hat']
replaceD([], {'cat':'dog', 'the':'that'}) --> []
```

(*6 pts*) 4. Write a Python3 function `nnn()` that satisfies the following spec, using recursion and no loops. (*Note:* "`nnn`" is short for "`nestedNegateNumbers`."

`nnn`
1 Argument: Any (possibly nested) list.
Return: A list consisting of the same atoms as *arg1* at the same levels of nesting, except with every number replaced by its negation.

Examples:
```
nnn([4, -1.5, "hi", [[6, "b"], 10]])
     --> [-4, 1.5, "hi", [[-6, "b"], -10]]
nnn([[6, "b"], 10])
     --> [[-6, "b"], -10]
nnn([]) --> []
```

(*6 pts*) 5. Write a `mapper()` and a `reducer()` for the WebMapReduce (WMR) framework that produces an index of words in a data set with line numbers and chapter names, satisfying the following spec. Assume that whitespace characters separate words in the text.

Mapper

```
# IN keys and values:
#   key: holds a line number, colon, then chapter name
#   value: a line of text
```

Reducer

```
# OUT keys and values:
#   key: holds a word from that text, in lower case
#   value: chapter name, space, line number, comma, and space for
#      each occurrence of that word in lines of input text
```

Example: If the data set is

```
1:Sam   I am Sam
2:cat   The cat in the hat
3:cat   wore the hat
```

then the resulting index will be

```
am      Sam 1,
cat     cat 2,
hat     cat 2, cat 3,
in      cat 2,
i       Sam 1,
sam     Sam 1,
the     cat 2, cat 2, cat 3,
wore    cat 3,
```

(*9 pts*) 6. a) Define a class `Firm` that has the state variables, methods, and constructor indicated in the accompanying spec (last page). **Include a comment that indicates the state variables.** *Note:* Only a class definition is required.

b) Define a class `Corp` that has the state variables, methods, and constructor indicated in the accompanying spec. Call the superclass constructor for `Firm` during initialization. *Note:* Only a class definition is required.

(*6 pts*) 7. Use turtle graphics to write Python3 code that draws a rectangle containing a word `Tools` as illustrated below. Include any `import` statements you need to make a complete program. *Note:* You only need to draw the rectangle and the word; labelled points are for illustration only. A list of method names is provided below.

- Two corners of the rectangle should be at coordinates $(25, 60)$ and $(75, 40)$.
- **The top of the rectangle should be drawn in red.** Everything else should be drawn in black
- The word `Tools` should appear inside the rectangle.

Common `turtle` names:

- `turtle.Screen()`, `turtle.Turtle()` `Screen` method `exitonclick()`
- Some methods of `Turtle`: `left()`, `right()`, `forward()`, `backward()`, `penup()`, `pendown()`, `goto()`, `color()`, `write()`

(*6 pts*) 8. Write a Python3 function `quarterGray()` that returns a modified copy of an image, satisfying the following spec.

`quarterGray`
1 Argument: A `cImage` image.
State change: A new `cImage` image with the same width and height as *arg1* is constructed, and each pixel in that new image is assigned the same pixel value as in *arg1*, except that pixels in the upper left quarter of the new image are grayscale pixels instead of copies.
Return: A `cImage` image, namely, that newly created and assigned `cImage` object.

Example call:

```
import cImage as image
win = image.ImageWin()
img = image.Image("greentree2.png")
img.draw(win)  # shows the original image
img2 = quarterGray(img)
img2.draw(win)  # shows the modified image
```

Hints:

- For a grayscale pixel, every color intensity (*red*, *green*, and *blue*) is replaced by the mean (average) of those color intensities. For example, a pixel with intensities (120,100,200) would become (140,140,140).
- Use the predefined function `int()` to convert any intensity values that might be non-integers into integers.
- A list of common `image`-related names is provided below.

Common `cImage` names:

- `image.EmptyImage(`*width*`,` *height*`)` `image.Pixel(`*red*`,` *green*`,` *blue*`)`
- Some methods of `image.Image`: `getWidth()`, `getHeight()` `setPixel(`*col*`,` *row*`,` *pixel*`)`

`Firm`

Instances represent: A business firm.

State variables:

`name`, The name of this firm.

`size`, The number of employees in this firm.

Constructor:

`Firm()`

2 Arguments: A string and a non-negative integer.

State change: A new object of type `Firm` is created. *arg1* is assigned to the state variable `name`, and *arg2* is assigned to the state variable `size`.

Return: Type `Firm`, namely, that object that was created.

Methods:

`getName(), getSize()`

0 Arguments.

Return: The value of the indicated state variable.

`setName()`

1 Argument: A string.

State change: The string *arg1* is assigned to the state variable `name`.

Return: The former value of the state variable `name`.

Example calls:
```
firm = Firm('Acme', 35)
firm.getSize() --> 35
firm.setName('Best') --> 'Acme'
firm.getName() --> 'Best'
```

`Corp`

Instances represent: An incorporated business firm.

Superclasses: `Firm`

State variables:

`id`, Integer id number for this firm.

`nextId`, Int, the id number for the next new firm, initially 1. (**CLASS VARIABLE**)

Constructor:

`Corp()`

2 Arguments: A string and a non-negative integer.

State change: A new object of type `Corp` is created. State variables inherited from `Firm` are initialized using *arg1* and *arg2*. `nextId` is assigned to the state variable `id`. Then, 1 is added to `nextId`.

Return: Type `Corp`, namely, that object that was created.

Methods:

`__str__()`

0 Arguments.

Return: A string that reports state variable values in the format

id. *name* (*size* `employees`)

Example calls:
```
corp = Corp("Acme", 35)
corp2 = Corp("Zenith", 490)
corp.getSize() --> 35
str(corp2) --> '2. Zenith (490 employees)'
```

Note: Solving 7 of the 8 problems above indicates the typical length of a final exam. The following are **additional problems** that illustrate other types of questions.

(*6 pts*) 9. For each of the following terms, (i) define that term in your own words, and (ii) illustrate that term using a Python 3 code example.

a) *function call*

b) *object*

c) *constructor*