**Carleton University**
**SYSC 1005 – Introduction to Software Development – Fall 2012**
**Midterm Exam – October 10, 2012**
**Version B**

**Name:**_____ANSWER KEY_____ _____

**Student Number:**_____

**INSTRUCTIONS:**

1. This exam is closed book. Calculators are not permitted.
2. Exam questions will not be explained, and no hints will be given. If you think something is unclear or ambiguous, make a reasonable assumption (one that does not contradict the question), write it at the start of your solution, and answer the question.
3. Remember, indentation is important in Python. You may want to draw light vertical lines to guide you when indenting your Python code.
4. The crib sheet on last two pages of this question paper summarizes some Python functions that you may find useful.
5. All pages of this question paper, including the crib sheet, must be turned in.

**Question 1 [10 marks]**

Here is an incomplete transcript from a session with the Python shell. On the ruled lines, write the values that Python would display after it evaluates each expression. If the expression would cause an error, write "Error". (You don't need to write the actual error message that Python would display.)

```
>>> -2 / 3 + 4
```

**3**

_____

```
>>> -3 ** 2
```

**-9**

_____

```
>>> (16-8.0) * (1/2)
```

**0.0**

_____

```
>>> str1 = """Hello """
>>> str2 = 'Class!'
>>> str3 = str1+str2
>>> str3
```

**'Hello Class!'**

---

```
>>> a = 5
>>> b = 2
>>> "There are " + (a+b) + " cans of tomatoes!"
```

**ERROR**

---

```
>>> a = 20
>>> is_less = a < 15
>>> if(is_less):
...     a=a+15
... else:
...     a=0
...
>>> a
```

**0**

---

```
>>> 5.0 // 2.0
```

**2.0**

---

```
>>> "Hello Canada!" - " Canada!" + " Ontario!"
```

**ERROR**

---

```
>>> 3 * 'Hi'
```

**'HiHiHi'**

---

```
>>> 0.5 * 'HiHi'
```

**ERROR**

---

**Question 2 [5 marks]**

Write a function called `calc_surface_area_cyl` that calculates and returns the surface area of a cylinder with one closed end. Below is a description of the mathematical calculation your function should perform. The thickness of the cylinder is negligible.

$$A = \pi r^2 + 2\pi rl$$

| | |
|---|---|
| *A:* | Surface area of the cylinder. |
| *r:* | Radius of either end of the cylinder. |
| *π:* | Pi |
| *l:* | Length of the cylinder. |

The function header is:

```
def calc_surface_area_cyl(r,l):
```

Parameter `r` is bound to the radius value of the cylinder ends, and `l` is bound to the value of the length of the cylinder.

**Write your function on the next page. Use the back of the page if you need more room.**

**Question 2 Solution**

```
import math
def calc_surface_area_cyl(r,l):
    return (math.pi*r**2)+(2*math.pi*r*l)
```

**Question 3 [10 marks]**

In the lab you wrote a function that created a greyscale image. Each pixel in the image was set to the average of its red, green, and blue intensity values.

Write a function called `make_green_monochrome` that sets **only** the green intensity of each pixel to the average of that pixel's initial red, green, and blue colour intensities. Your function should also set the red and blue intensities of each pixel to zero.

The function header is:

```
def make_green_monochrome(pict):
```

Parameter `pict` is bound to a `Picture` object when the function is called.

**Write your function on the next page. Use the back of the page if you need more room.**

**Question 3 Solution**

```
def make_green_monochrome(pict):
    for pixel in CU_media.get_pixels(pict):
        red = CU_media.get_red(pixel)
        green = CU_media.get_green(pixel)
        blue = CU_media.get_blue(pixel)
        ave = (red+green+blue)/3
        CU_media.set_red(pixel,0)
        CU_media.set_green(pixel,ave)
        CU_media.set_blue(pixel,0)
```

**Crib Sheet**

**Built-in Python functions:**

`abs(`*`x`*`)`
: Return the absolute value of *`x`* (an `int` or a `float`).

`float(`*`x`*`)`
: Convert argument *`x`* (a string or number) to a real number, if possible.

`int(`*`x`*`)`
: Convert argument *`x`* (a string or number) to an integer, if possible. A real number argument will be truncated towards zero.

`max(`*`a, b, c, ...`*`)`
: With two or more arguments, return the largest argument.

`min(`*`a, b, c, ...`*`)`
: With two or more arguments, return the smallest argument.

`range(`*`stop`*`)`
: Return the list of integers `[0, 1, 2, ..., `*`stop`*` - 1]`.

`range(`*`start, stop`*`)`
: Return the list of integers `[`*`start`*`, `*`start`*` + 1, `*`start`*` + 2, ..., `*`stop`*` - 1]`.

`raw_input(`*`prompt`*`)`
: Read a string from the keyboard and return the string. The *`prompt`* string, if provided, is printed without a trailing newline before reading the string.

`round(`*`x, n`*`)`
: Return the `float` that results when the value of argument *`x`* (a `float`) is rounded to *n* digits after the decimal point.

`str(`*`x`*`)`
: Return a printable string representation of argument *`x`*.

**`math` module:**

`pi`
: High precision constant for Pi.

`cos(x)`
: Return the cosine of x (measured in radians).

`sin(x)`
: Return the sine of x (measured in radians).

`tan(x)`
: Return the tangent of x (measured in radians).

`atan2(y,x)`
: Return the arc tangent (measured in radians) of y/x.
Unlike atan(y/x), the signs of both x and y are considered.

**`CU_media` module:**

`choose_file()`
: Launch a file chooser and return a string containing the name of the file that was selected.

`load_picture(`*`filename`*`)`
: Create a `Picture` object from the contents of *`filename`* (a string) and return it.

*This crib sheet continues on the following page.*

**Functions that work with `Picture` objects:**

`show_external(`*`pict`*`)`
: Display the picture.

`get_height(`*`pict`*`)`
: Return the height of the picture in pixels (an `int`).

`get_width(`*`pict`*`)`
: Return the width of the picture in pixels (an `int`).

`get_pixels(`*`pict`*`)`
: Return a list containing all the pixels in the picture.

`get_pixel(`*`pict, x, y`*`)`
: Return the pixel at the specified (*`x`*, *`y`*) location in the picture.

**Functions that work with `Pixel` objects:**

`get_red(`*`pixel`*`)`, `get_green(`*`pixel`*`)`, `get_blue(`*`pixel`*`)`
: Return the values of the pixel's red, green and blue components (an `int` between 0 and 255).

`set_red(`*`pixel, amount`*`)`, `set_green(`*`pixel, amount`*`)`, `set_blue(`*`pixel, amount`*`)`
: Set the values of the pixel's red, green and blue components to the specified *`amount`* (which must be an `int` between 0 and 255).

`get_color(`*`pixel`*`)`
: Return a `Color` object that represents the colour of the pixel.

`set_color(`*`pixel, color`*`)`
: Set the pixel's colour to the specified *`color`* (a `Color` object).

`get_x(`*`pixel`*`)`
: Return the x-coordinate of the pixel's location.

`get_y(`*`pixel`*`)`
: Return the y-coordinate of the pixel`s location.

**Functions that work with `Color` objects:**

`Color(`*`red`*`, `*`green`*`, `*`blue`*`)`
: Return a `Color` object with the specified amounts of *`red`*, *`green`* and *`blue`*. (Each amount must be an `int` between 0 and 255).

`distance(`*`color1, color2`*`)`
: Return the distance between the `Color` objects bound to *`color1`* and *`color2`*.

`get_red(`*`color`*`)`, `get_green(`*`color`*`)`, `get_blue(`*`color`*`)`
: Return the values of the color's red, green and blue components (an `int` between 0 and 255).