PASS MOCK EXAM – *FOR PRACTICE ONLY*

Course: SYSC 1005A Facilitator: Veronica Santos

Dates and locations of mock exam take-up: Dec.11, 2013, 12:00 -15:00, ME 3275

IMPORTANT:

It is **most beneficial** to you to write this mock midterm **UNDER EXAM CONDITIONS**. This means:

- Complete the exam in 3 hour(s).
- Work on your own.
- Keep your notes and textbook closed.
- Attempt every question.

After the time limit, go back over your work with a different colour or on a separate piece of paper and try to do the questions you are unsure of. Record your ideas in the margins to remind yourself of what you were thinking when you take it up at PASS.

The purpose of this mock exam is to give you practice answering questions in a timed setting and to help you to gauge which aspects of the course content you know well and which are in need of further development and review. Use this mock exam as a ***learning tool*** in preparing for the actual exam.

Please note:

- Come to the PASS session with your mock exam complete. There, you can work with other students to review your work.
- Often, there is not enough time to review the entire exam in the PASS session. Decide which questions you most want to review – the Facilitator may ask students to vote on which questions they want to discuss.
- Facilitators do not bring copies of the mock exam to the session. Please print out and complete the exam before you attend.
- **Facilitators do not produce or distribute an answer key for mock exams.** Facilitators help students to work together to compare and assess the answers they have. If you are not able to attend the PASS session, you can work alone or with others in the class.

Good Luck writing the Mock Exam!!

Question 1 [10 marks]

Below 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).

```
>>> 12 + 18 / 4 * 4.0 - 6
```

_____

```
>>> santa = "Ho " * 3
>>> santa
```

_____

```
>>> lst = [1,2,34,50]
>>> i = lst[1]
>>> i
```

_____

```
>>> a1 = {3,5,3,8}
>>> a2 = {90,42}
>>> a3 = a2 | a1
>>> a3
```

_____

```
>>> b = 3
>>> C = 3/2
>>> b,c = c, b
>>> c
```

_____

```
>>> things = {(2,34),(0,'blue',5),(1,79)}
>>> things[2]
```

_____

```
>>> p = 8,3,12,4
>>> l = list(p)
>>> q,r,t,y = p
>>> q/3*l[2]+r**(y/2)
```

```
>>> animals = {“Dog”:3, “Frog”:1, “Cat”:2}
>>> animals [“Dog”] = 1
>>> animals [“Dog”]
```

```
>>> f = “Apple”
>>> z = “ pie!”
>>> len(f+z)
```

```
>>> num1 = 23
>>> num2 = num1 + 8.0
>>> num1 = num2 + 10
>>> num1
```

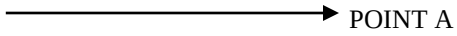
Question 2 [30 marks]

Below are two pieces of Python code. When executed, these codes do not produce an error.

a)

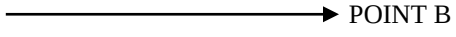
```
def count(lst, num):

    count = 0
    for i in lst:
        if i == num:
            count = count + 1
    return count
```



POINT A

```
lst = [2,1,2,55,7,2,1]
b = 2
total = count(lst, b)
```



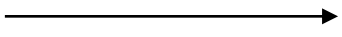
POINT B

i) Draw the memory diagram of the above code up to POINT A

ii) Draw the memory diagram of the above code up to POINT B

b)
```
def something(t1):

    x, y, z = t1
    z, y = y, z
    total = x * z + y
    return total
```



POINT A

```
this = 2,3,4
answer = something(this)
```


POINT B

i) Draw the memory diagram of the above code up to POINT A

ii) Draw the memory diagram of the above code up to POINT B

Question 3 [20 marks]

Write a function called `words_in_text,` this function takes a text file as the argument. This function should return the word in the text with most occurrences (for example, *The* and *the* should be considered as the same word). In addition, a listing of all the unique words in the text file should be returned in alphabetical order with all punctuation removed and in lower case.

The function header should be:

```
words_in_text (filename):
```

Use the space below or the back of the page to write your function.

Question 4 [20 marks]

Throughout the semester you have created many different filters to manipulate images in Python using the Cimpl module. Sometimes pictures may be corrupted by noise, causing the picture to be unclear or fuzzy. To reduce this noise and produce a smoother image, the red, green, and blue components of each pixel in the image is replaced by the *median* red, green, and blue components of the pixel and its neighbouring pixels (the pixels above, below, to the left, and to the right). The *median* is the middle value obtained by sorting the colour components of the pixel and its neighbouring pixels in numerical order. For example, if the red component values of the 5 pixels are 40, 88, 101, 178, 234, the median is 101. Create a filter called `reduce_noise`, the filter's argument is an image object. The filter returns a smoother image with the noise removed.

The function header should be:

```
def reduce_noise (img):
```

For simplicity, the filter should ignore the edges of the image. A person viewing the modified image would not notice the difference.

Use the space below or the back of page to write your function.

Question 5 [20 marks]

The game of craps is a popular chance game played all over the world. The rules are simple, the game starts with the player rolling a pair of dice. If the player rolls a total of 7 or 11, the player wins. If the player rolls a total of 2, 3, or 12, the player loses. For all other rolls the game keeps going until the next roll is the same as the initial roll (the player wins in this case) or the roll is a 7 (the player loses in this case). Create a function called `craps`, this function takes no arguments. The function returns a 1 if the player wins or it returns a 0 if the player loses. Include print statements to inform the player if he or she won or lost the game. Keep in mind that the rolls are random.

The function header should be:

```
def craps ():
```

Use the space below or the back of page to write your function.

Question 6 [10 marks]

Convolution is a technique commonly used in image processing to improve the outcome of common effects. Below on the left is a convolution kernel used to produce an edge-detection effect, and on the right is an image with each square representing a pixel. The RGB components of certain pixels are also shown.

Convolution Kernel

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Image

Pixel 1

r = 120 g = 34 b = 211	r = 30 g = 49 b = 199	r = 252 g = 111 b = 97			
r = 76 g = 201 b = 76	r = 31 g = 72 b = 13	r = 18 g = 60 b = 55			
r = 3 g = 93 b = 242	r = 90 g = 98 b = 61	r = 72 g = 233 b = 1			

Using the convolution kernel technique to give *Pixel 1* an edge-detection effect, calculate the new red, green, and blue components of *Pixel 1*.