

THE UNIVERSITY OF MANITOBA

COMP1012:  
Computer Programming for Scientists and Engineers  
Final Exam (3 hours)

Name
Student number

Section (please check one):

- ☐ A01 (Andres EITC E2-110)
- ☐ A02 (Melvin Buller 207)

December 12, 2013, 9:00 am

MARKS

Exam Instructions:

- Place your student card on your desk.
- Answer all questions.
- No aids are permitted.
- Write your name, student number, and section on all *separated* pages.
- Write all of your answers on the exam.
- Marks add up to 50.

For Graders Only:

A:     \_\_\_\_\_/10  
B:     \_\_\_\_\_/25  
C:     \_\_\_\_\_/10  
D:     \_\_\_\_\_/ 5  
Total: \_\_\_\_/50

Part A: Predict the output (10 × 1 MARK)

There is a separate problem in each row of the table below. In each one, mentally execute the code fragment on the left and enter the expected output in the box on the right. *None result in an error.* Use the last page of the exam for scrap work.

	Code Fragment	Expected output
1. [1]	<code>print 2**3 // 5 % 4</code>	
2. [1]	<code>print tuple(np.linspace(1,2,2))</code>	
3. [1]	<code>print 1 &lt;= 1 &gt;= 2**1</code>	
4. [1]	<code>print range(-2,-1,2)</code>	
5. [1]	<code>print 'alphabet'[-5:5]</code>	
6. [1]	<code>print ('Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun')[-5:2]</code>	
7. [1]	<code>print [jj + 5 // 3 for jj in [0, 5, 3, 3] ]</code>	
8. [1]	<code>print (array([3, 3, 0, 4]) * 2 - 2)</code>	
9. [1]	<code>print "123" + "877"</code>	
10. [1]	<code>print 'The quick brown fox'.split()</code>	

MARKS

Part B: Programming – Full Program (4 + 2 + 7 + 4 + 6 + 2 = 25 MARKS)

A ‘vicious circle’ is a shape where half the points lie on an inner circle and half lie on an outer circle. Write a program that reads input about a vicious circle, then plots it, and finds its area. Below is the printed output from one run of the program, and the plotted output is shown below that on the left.

Analyzing a Vicious Circle

Enter the number of teeth in a vicious circle: 30

Enter the ratio of outer to inner radius (between 1 and 2): 0.8

Out of range; enter a value between 1 and 2.

Enter the ratio of outer to inner radius (1 to 2): 1.2

Area with inner & outer radii 10 & 12 [cm] and 61 points is 376 [cm^2]

Program terminating

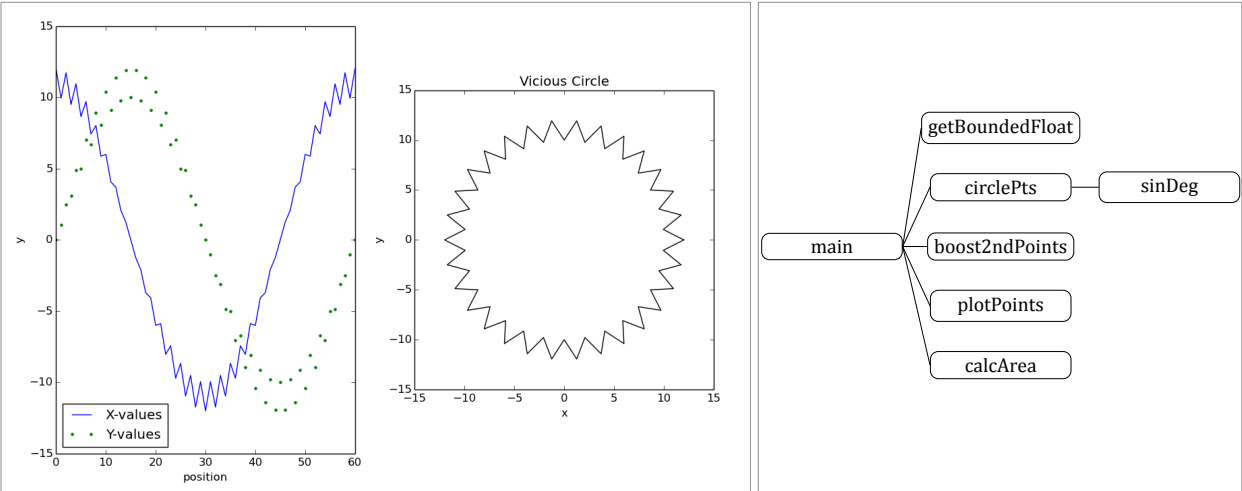


Figure 1 : Left: Program Plot Output; Right: Structure Chart

The main function has been programmed for you (below). Study it carefully. All of the interesting work done by the program is performed in the functions it calls. You will write the five underlined functions in main (no choice), and also `sinDeg`, used at a lower level. (See the structure chart above right.) Each function will be marked separately—even if you have problems with one, you can get full marks on others.

```
import matplotlib.pyplot as plt
import numpy as np
DEGREES_IN_CIRCLE = 360.
def main() :
    print "Analyzing a Vicious Circle"
    numTeeth = int(getBoundedFloat(3, 100,
        "Enter the number of teeth in a vicious circle: "))
    numPoints = 2 * numTeeth + 1
    ratio = getBoundedFloat(1.0, 2.0,
        "Enter the ratio of outer to inner radius "
        "(between 1 and 2): ")
    radius = 10.
    points0 = circlePts(radius, numPoints)
    points1 = boost2ndPoints(points0, ratio)
    plotPoints(points1, "Vicious Circle")
    area = calcArea(points1)
    print ("\nArea with inner & outer radii %g & %g [cm] "
        "and %d points is %g [cm^2]"
        % (radius, radius * ratio, numPoints, round(area)) )
    print "\nProgram terminating"
    return
```

Do NOT put any answers on this page.

Name

MARKS

B.1    Function `getBoundedFloat` [4 MARKS]

[4]    Define a function `getBoundedFloat(loBnd, hiBnd, prompt)` that matches the call from `main`. The parameters `loBnd` and `hiBnd` are lower and upper bounds on the value the user can enter. The user can enter either of these values or anything in between. The parameter `prompt` is the string that this function shows the user to get input. The input is a single float value. This function loops as many times as it takes to get a valid input from the user, which it returns as a float to the calling function.

Assume the user always enters a single float value, so you don't have to check for non-float input. You just have to check that the input's value lies in the correct interval. If the user enters an unacceptable value, print a warning message before prompting again. The warning message must contain `loBnd` and `hiBnd`, formatted. For example, entry of the ratio with a user mistake might look like this:

Enter the ratio of outer to inner radius (1 to 2): 0.8

Out of range; enter a value between 1 and 2.

Enter the ratio of outer to inner radius (between 1 and 2): 1.2

```
def getBoundedFloat(loBnd, hiBnd, prompt) :  
    """Ask the user for a float between loBnd and hiBnd using prompt,  
    and keep asking till you get a valid float. Then return it."""
```

For marker use only	
Item	Mark
A	
B	
C	
D	
E	
Sum	

*Note: there is extra space on page 8*

Name

MARKS

B.2    Function circlePts [2 MARKS]

- [2]
- Define a function circlePts(radius, numPoints) that matches the call from main.
- It returns a 2-element tuple containing a numpy array with numPoints  $x$  points and a numpy array of numPoints  $y$  points. Pairs of  $(x,y)$  points from these two arrays are spaced evenly along the circumference of a circle with the given radius, centred at  $(0,0)$ . Points at **both  $0^\circ$  and  $360^\circ$  are included**.
  - The formulas for  $x$  and  $y$  are:

$$x = r \cos \theta_j$$
$$y = r \sin \theta_j$$
where  $\theta_j$  is one of the equally spaced angles from  $0^\circ$  to  $360^\circ$ .
  - Call the functions sinDeg and cosDeg to evaluate sin and cos functions (**not np or math functions**). These functions take **array arguments in degrees**, not radians. You will program sinDeg in B.3. Assume cosDeg is also available.
  - Note: in this and the following functions, you must accept and/or return numpy arrays, but you may use array calculations or not, as you please.**
- ```
def circlePts(radius, numPoints) :  
    """Return arrays of x values and y values from numPoints points  
    evenly spaced from 0 to 360 degrees along the circumference of a  
    circle with given radius, centred at the origin (0,0)."""
```

| For marker use only |      |
|---------------------|------|
| Item                | Mark |
| A                   |      |
| B                   |      |
| C                   |      |
| D                   |      |
| E                   |      |
| Sum                 |      |

Note: there is extra space on page 8

Name

MARKS

B.3    Function sinDeg [7 MARKS]

[7]    Define a function `sinDeg` that matches the call from `circlePts`. It receives a single parameter: `degrees`, which is a *numpy array of angles* in units of degrees. It converts these angles to radians, then evaluates the sine function for each angle using a power series. It returns the array of sine values corresponding to the incoming `degrees`.

The following formula transforms an incoming angle  $x_{\text{degrees}}$  in degrees into a ‘reduced’ angle in radians between  $-\pi$  and  $\pi$ , that has the same sine value as the incoming angle. The expression “ $a \bmod 360$ ” finds the *float* remainder when dividing  $a$  by 360.

$$x_{\text{rad}} = ((x_{\text{degrees}} + 180) \bmod 360 - 180) \frac{\pi}{180}$$

Evaluate the following series for  $\sin(x)$ :

$$\sin(x) = \frac{x}{1!} - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} \cdots$$

*Use exactly 15 terms* of the series (that is, terms up to  $x^{29}/(29!)$ ). (Because every reduced angle  $x_{\text{rad}}$  lies between  $-\pi$  and  $\pi$ , the series converges with 15 terms for any incoming angle.) *Do not use any factorial function.*

```
def sinDeg(degrees) :  
    """degrees is an array of angles in degrees. Return an array  
    of sine values, one for each angle."""
```

| For marker use only |      |
|---------------------|------|
| Item                | Mark |
| A                   |      |
| B                   |      |
| C                   |      |
| D                   |      |
| E                   |      |
| Sum                 |      |

*Note: there is extra space on page 8*

Name

MARKS

B.4    Function `boost2ndPoints` [4 MARKS]

[4]

Define a function `boost2ndPoints(pointsIn, factor)` that returns a new tuple similar to `pointsIn`, except that every second point has been multiplied by `factor`.

- `pointsIn` is a 2-element tuple `(xs, ys)` containing two numpy arrays of the same length; corresponding values from `xs` and `ys` are assumed to make  $(x,y)$  points.
- Use an `assert` to check that the two arrays in `pointsIn` are the same length.
- Use the following formula to calculate the  $x$  and  $y$  points to return in a tuple, where  $a$  represents the incoming `factor`; **every 2<sup>nd</sup> point**  $(x,y)$  in `pointsIn` is boosted:

$$(x'_j, y'_j) = \begin{cases} (x_j, y_j) & \text{if } j \text{ is odd} \\ (ax_j, ay_j) & \text{if } j \text{ is even} \end{cases}$$

- Make sure you do not change values in `pointsIn`.***

```
def boost2ndPoints(pointsIn, factor) :  
    """pointsIn should be a tuple with two arrays, one of x-values  
    and one of y-values, of the same length. factor is a number.  
    Multiply every second point by factor, and return the modified  
    collection of points, without changing pointsIn."""
```

| For marker use only |      |
|---------------------|------|
| Item                | Mark |
| A                   |      |
| B                   |      |
| C                   |      |
| D                   |      |
| E                   |      |
| Sum                 |      |

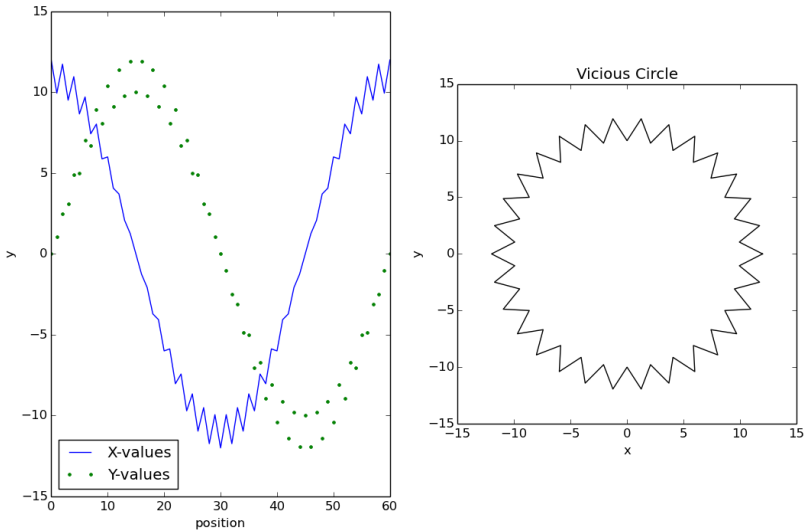
*Note: there is extra space on page 8*

MARKS

B.5    Function plotPoints [6 MARKS]

- [6]
- Define a function `plotPoints(points, title)` that matches the call from `main`, to plot 2 views of a set of points in 2 subplots of a single figure, as shown below.

  - `points` is a 2-element tuple `(xs, ys)` containing two `numpy` arrays of the same length; corresponding values from `xs` and `ys` are assuming to make `(x,y)` points.
  - Draw a figure with two subplots, side-by-side. They must both have axis labels.
  - The left subplot shows `x` and `y` values plotted separately against point number. One line must use a solid curve, and the other shows only points. A legend is required.
  - The right subplot shows `y` values plotted against `x` values. It uses `title` as a title.



```
def plotPoints(points, title) :  
    """points should be a tuple with two arrays, one of x-values  
    and one of y-values, of the same length. Prepare two side-by-  
    side subplots, one showing the x- and y-values separately,  
    and one of y-values plotted against x-values."""
```

| For marker use only |      |
|---------------------|------|
| Item                | Mark |
| A                   |      |
| B                   |      |
| C                   |      |
| D                   |      |
| E                   |      |
| Sum                 |      |

|      |
|------|
| Name |
|------|

MARKS

*Extra space for answering any part of the programming question. There is also room on the last page. Keep going; there are more questions ahead.*



Name

MARKS

B.6    Function calcArea [2 MARKS]

[2]    Define a function calcArea(points) that matches the call from main, to calculate the area defined by the provided points.

- points is a 2-element tuple (xs, ys) containing two numpy arrays of the same length; corresponding values from xs and ys are assuming to make (x,y) points.
- the area defined by two consecutive points (xi, yi) and (xi+1, yi+1) is the trapezoidal area:

$$A_i = \frac{1}{2}(x_{i+1} - x_i)(y_{i+1} + y_i)$$

- do **not** try to join the last point back to the first; if points defines a closed curve, the last point will already be identical to the first.

```
def calcArea(points) :  
    """points should be a tuple with two arrays, one of x-values and  
    one of y-values, of the same length. Calculate its area using  
    trapezoidal integration and return it."""
```

| For marker<br>use only |      |
|------------------------|------|
| Item                   | Mark |
| A                      |      |
| B                      |      |
| C                      |      |
| D                      |      |
| E                      |      |
| Sum                    |      |

**COMP1012:**  
**Computer Programming for Scientists and Engineers**  
**Final Exam (3 hours)**

MARKS

**Part C: Multiple choice (10 x 1 MARK)**

For each of the following 10 multiple-choice questions, circle the *single best* answer.

- [1] 1. Which of the following formatted prints would produce the given output?
- Output: | -0.2857 |
- a) `print "Output: |%-20.4e|" % (-2./7)`
  - b) `print "Output: |%-20.4f|" % (-2./7)`
  - c) `print "Output: |%-20g|" % (-2./7)`
  - d) `print "Output: |%20.4f|" % (-2./7)`
  - e) `print "Output: |%20r|" % (-2./7)`
- [1] 2. Which of the following statements about while loops is *not* correct?
- a) If a while loop begins as follows, it will quit looping immediately:  
`while 'False' :`
  - b) When a while statement is executed, the indented instructions in the loop may be executed zero times.
  - c) Suppose statements in a given while loop are executed at least once. If the loop begins as follows, code in the loop must change the value of variable `stillGoing`, or the loop will never stop.  
`while stillGoing :`
  - d) Indented instructions in a while loop are executed an integer number of times.
  - e) A for loop is preferred to a while loop when looping a known number of times.
- [1] 3. Which of the following could be the result of the statement  
`print np.random.randint(-2,6,4) ?`
- a) `[ 3 -1 2 2 0]`
  - b) `[ 6 -1 2 0]`
  - c) `[ 3 -1 2 -2 1 2]`
  - d) `[ 5 -1 2 2]`
  - e) `[ 4 -1 2 0 2 3]`
- [1] 4. After the following code executes what is the value of `yVals`?
- ```
def evalPoly(poly,xVals) :
    xVals = np.array(xVals)
    fxVals = np.zeros(len(xVals))
    for coef in reversed(poly):
        fxVals = fxVals * xVals + coef
    return fxVals

coef = [2,2,2]
xVals = np.array([1,2,3])
yVals = evalPoly(coef,xVals)
```
- a) `array([ 2., 11., 12.])`
  - b) `array([ 6, 14, 26 ])`
  - c) `array([ 2., 2., 2.])`
  - d) `array([ 6., 14., 26.])`
  - e) `array([ 17, 17, 17])`

COMP1012:  
Computer Programming for Scientists and Engineers  
Final Exam (3 hours)

MARKS

- [1]

5. Which of the following statements is correct?

a) Lists and tuples are immutable, but strings and numpy arrays can change.

b) Lists and numpy arrays are immutable, but tuples and strings can change.

c) Tuples and strings are immutable, but lists and numpy arrays can change.

d) Strings and lists are immutable, but tuples and numpy arrays can change.

e) Tuples, strings and numpy arrays are immutable; only lists can change.
- [1]

6. After the following sequence of instructions is executed, what is the value of letters?

```
letters = "ABCDEFGH"  
letters = letters[-1: :-1]  
letters = letters[:3][-1: :-1] + letters[3:][-1: :-1]
```

a) "ABCDEFGH"

b) "CBAGFED"

c) "EFGABCDEFGH"

d) "EFGABCD"

e) ""
- [1]

7. If the numbers in the following structure charts represent the order in which you implement the corresponding functions (e.g., you always implement function 5 before function 6), which of these charts represents top-down programming?

(a)

```
graph TD; 0 --> 1; 0 --> 2; 0 --> 4; 0 --> 5; 1 --> 7; 1 --> 8; 2 --> 3; 5 --> 9; 5 --> 10; 10 --> 6;
```

(b)

```
graph TD; 5 --> 1; 5 --> 3; 5 --> 6; 5 --> 8; 1 --> 0; 1 --> 2; 3 --> 4; 8 --> 7; 8 --> 9; 9 --> 10;
```

(c)

```
graph TD; 10 --> 2; 10 --> 4; 10 --> 5; 10 --> 9; 2 --> 0; 2 --> 1; 4 --> 3; 9 --> 7; 9 --> 8; 8 --> 6;
```

(d)

```
graph TD; 0 --> 1; 0 --> 4; 0 --> 6; 0 --> 10; 1 --> 2; 1 --> 3; 4 --> 5; 10 --> 9; 10 --> 8; 8 --> 7;
```

(e)

```
graph TD; 0 --> 1; 0 --> 4; 0 --> 6; 0 --> 7; 1 --> 2; 1 --> 3; 4 --> 5; 7 --> 8; 7 --> 9; 9 --> 10;
```

- [1]

8. What is meant by Horner's Method?

a) A method for efficiently evaluating a polynomial.

b) A random number generator used by Python.

c) A way of determining all prime numbers up to a limit.

d) A rule describing the exponential growth of computer chip capacity.

e) The best way of summing up an infinite series.

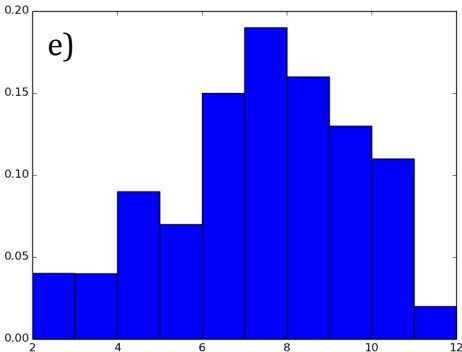
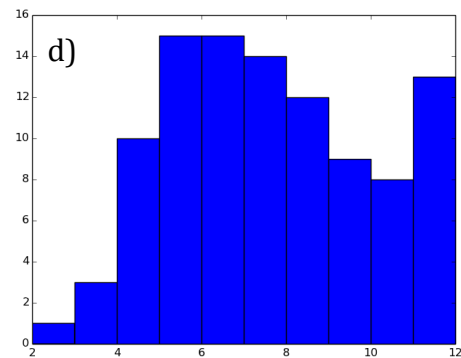
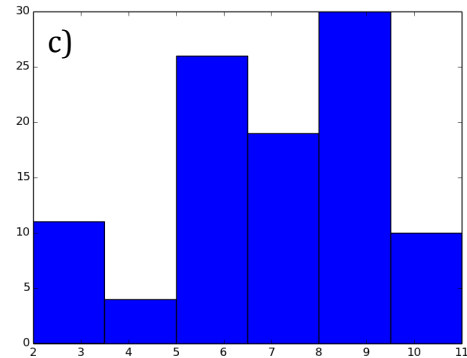
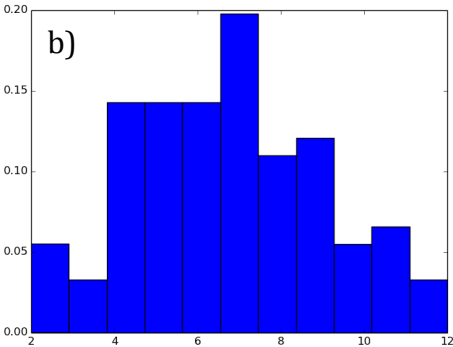
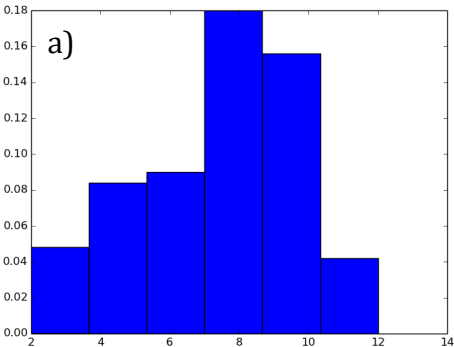
Name

Marks

[1]

9. Which plot could result from the following code?

```
import numpy as np
import matplotlib.pyplot as plt
xx1 = np.random.randint(1,7,100)
xx2 = np.random.randint(1,7,100)
dice = xx1 + xx2
plt.hist(dice, normed=True)
plt.show()
```



10. [1 marks] Using good coding practices, and the same rules as QuizMaster write a Python expression to evaluate this mathematical expression, assuming math has already been imported:

$$-\sqrt{\ln\left(\sqrt{5}\right)}\cdot\left(10^{\frac{a}{\ln(10^4)}}\right)$$

Put expression here

Name

MARKS

Part D: Short Answer (5 MARKS)

[5] The code below prints successive versions of a user’s word, with hidden letters gradually revealed one letter at a time. Sample output is in the box at the right.

The code was originally correct, but 6 errors have been added to it. Each error is one of the following: *missing/extra/incorrect item, where item is punctuation, constant, operation, library, function or variable* (including mismatched brackets or quotes). Only numbered lines have errors, and no line has more than one error. Find the errors. In the table at the bottom, give the line number of each error, say what is wrong and how you would fix it. As an example, one error has been done for you. Find five more.

```
1 def showWord(word, guessed)
    """Create a string showing the word double-spaced,
    with unguessed letters represented by underscores"""
2     display = ''
3     for letter in guessed :
4         if letter in guessed :
5             display += letter + ' '
6         else :
7             display = "_ "
8     return letter

9 word = raw_input("Enter a word: ").strip().upper()
10 letters, guessed = word, '', 3
11 while letters :
12     guessed = guessed + letters[0]
13     letters = letters.replace(letters[0] : '')
14     print showWord(word, guessed)
```

Sample output:  
Enter a word: bananas  
B \_ \_ \_ \_ \_  
B A \_ A \_ A \_  
B A N A N A \_  
B A N A N A S

(1) line 1... ‘:’ is missing at the end the **def**; add it after the closing parenthesis’)

(2)

(3)

(4)

(5)

(6)

Name
------

MARKS

*Extra space*

Name

MARKS

*Extra space*

