THE UNIVERSITY OF MANITOBA
FINAL EXAMINATION

DATE: December 19, 2017, 6 pm **PAGES:** Page **1** of **12**
DEPT. & COURSE #: COMP 1012 **TIME: 3 hours**
EXAMINATION: Computer Programming for Scientists and Engineers
EXAMINERS: Boyer, Pazdor

Name:_____ **Student No:**_____

Section (please circle one): A01 (Boyer) A02 (Pazdor)

MARKS

Exam Instructions:

- Marks add up to 50.
- Several parts of the exam have bonus questions, each worth one mark. **The bonus mark may be used to make up a lost mark in that part of the exam only**, but will ***not*** count towards lost marks in other parts of the exam.
- No aids are permitted. Put all electronic devices away.
- Answer all questions, and write your answers on the exam itself.
- Write your name, student number, and section on this page and any ***separated*** pages.
- *Place your student card on your desk.*

For Graders Only:

A: _____/10

B: _____/10

C: _____/10

D: _____/10

E: _____/10

Total: _____/50

Part A: Predict the output [10 × 1 MARK + bonus]

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.	`print(-3**2 % 4)`	
2.	`print(2 == 1 or 3 and 4)`	
3.	`print("{0:}{1:}".format(20, "\N{DEGREE SIGN}"))`	
4.	`foo = [xx % 3 == 0 for xx in range(1, 10, 2)]` `print(foo)`	

5.	`import numpy as np`
	`foo = np.resize(np.arange(6), (2,3))`
	`print(foo[1])`	

6.	`import numpy as np` `print(np.array([1,2,3,4])*np.array([2,0,3,2]))`	
7.	`print("mississippi"[-9:-1:2])`	
8.	`print(5 % 2 * 'python')`	
9.	`print("*{0:^8.4f}*".format(1 / 3))`	
10.	`bb = set(list('mississippi'))` `print(len(bb))`	
Bonus	`print('{0:.{1:d}f}'.format(1/3, 8))`	

MARKS

Part B: B.1 + B.2 + B.3 form one program [3 + 5 + 2 = 10 MARKS]

Write a program that reads tabular data from a text file. The text file has a header line, followed by data lines. The file includes both continuous (X-Y values as floats) data and discrete (integer) data. The data is then plotted in two separate plots: an X-Y plot for the continuous data and a histogram for the discrete data.

B.1 Function readData [3 MARKS]

This function accepts one parameter: a string containing the name of a text file (such as `data.csv`). The file contains tabular data in comma-separated value format. It opens that file and reads the header information and data. The header is a tuple of strings, and the data is a list of lists. Continuous data must be stored as float values, and discrete data must be stored as integer values. In order to tell which column is integer data, the header for that column contains the string `'discrete'`.
The function returns the headers and data.

Some of the code is provided. Complete the function by providing the missing parts.

[3]

```
def readData(filename) :
    """Read tabular data from a file. Assume the file contains a header line
     followed by data lines. Two columns are float data, and the third
     column is integer data. Return the headers and data."""

    infile = open(filename, 'r')
    headers = infile.readline().strip().split(',')
    data = []
    for item in headers:
        data.append([])

# ADD YOUR CODE HERE

    infile.close()
    return headers, data
```

MARKS

Part B: B.1 + B.2 + B.3 form one program [3 + 5 + 2 = 10 MARKS]

B.2 Function plotData [5 MARKS]

This function produces two plots of the data read by the previous function. Each plot is produced in a separate figure. This function accepts three parameters; the headers, the data, and a dictionary. The dictionary contains one item for each of the headers. The value of each item is the index of the column in `data` for that header.
For example, if the header line in the data file was **"X,Y,discrete"**, the dictionary will contain the following items: {**"X": 0, "Y": 1, "discrete":2**}.

The first plot is a plot of the continuous data. Label the axes using the header information, and add a title. No legend is required.

The second plot is a normed histogram of the discrete data. Use the dictionary to determine which column in `data` contains the `discrete` data. Add appropriate axis labels and title. Make sure you produce one bin for each value the range of values (e.g. if the values range from 10 to 20 you should have 11 bins). DON'T hard-code 11 into your answer!

[5]

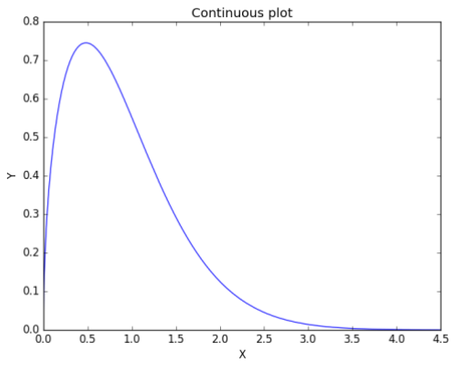
```
def plotData(headers, data, dataDict) :
    """Produce two plots. One is a plot of continuous data. The second
    is a histogram of discrete data. """
```

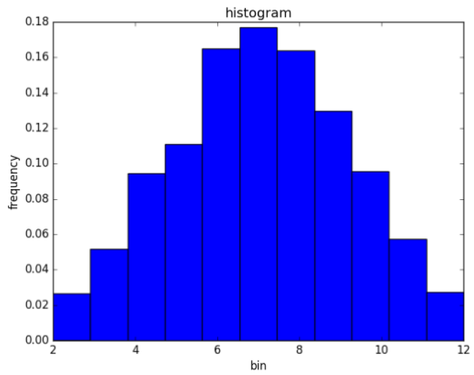
MARKS

Part B: B.1 + B.2 + B.3 form one program [3 + 5 + 2 = 10 MARKS]

B.3 Function main [2 MARKS]

Complete the code for the `main` function below. Prompt the user for the name of the data file. Call readData. Using the headers returned, produce a dictionary with one item for each of the values in the headers (as described in the previous question). Call plotData to produce the plots shown. Print an end of processing message.





[2]

```
def main( ) :
    """The main function"""
```

MARKS

Part C: Individual Programs

C.1 Utility Functions [3 x 2 MARKS + 1 bonus mark]

On this page, four function headers are provided. Choose **two** and implement them **on the next page**. You may assume without checking that all parameters are valid (i.e. no one has given you an integer where you need a string, and so on). Unless specifically noted in the function, you are **not permitted** to use any python libraries (such as math, numpy, or statistics).

```
def averageIf(collection, condition) :
    """This function returns the average value of all the items in
collection that meet the given condition. The parameter condition will be a
function that returns either True or False. This function should call
condition on each value in collection and then return the average of the
values for which condition returns True. If collection is empty or contains
no entries that meet condition, then this function returns 0."""
     #example: averageIf([0,1,2,3], bool) returns 2 (the average of 1,2,3)
     #because bool(0) is False but bool(x) is True for x = 1, 2, or 3

def shift(data, num) :
    """Returns a copy of data, where the entries of data have been shifted
to the left by num places. This function should work on collections (lists,
strings, tuples) as well as integers, but not floats"""
     #Note: this function is very easy for collections, but making it work
     #for integers will take more work
     #example: shift("smile",1) returns "miles"
     #example: shift([1,2,3,4,5], 3) returns [4, 5, 1, 2, 3]
     #example: shift(1234, 2) returns 3421

def gcf(numList) :
    """Returns the greatest common factor of the numbers in numList. That
is, it returns the largest integer X such that X evenly divides every
number in numList. There is a well-known algorithm for doing this, but you
may use any method you wish."""

def blackjack(c1, c2) :
    """Accepts two cards (two-character strings, like '5H' for the five of
hearts). Returns two things: the total of the card values and a string that
is either empty or contains the word 'Blackjack' if the total is 21."""
     #Note: Jacks, Queens, and Kings are worth 10 points
     #Aces are worth 1 unless the other card is worth 10,
     #in which case they are worth 11.
```

MARKS

C.1 Utility Functions

[6] Implement your chosen functions in the space provided here, ***including reproducing the header from the previous page.*** You do not need to reproduce the doc string or any comments.

Bonus: If you choose, you may implement three functions instead of just two. If you do so, the first two functions will be for marks, and the third will be given one bonus mark if it is correct.

MARKS

C.3 FizzBuzz [4 MARKS]

Write a program that will play the children's game FizzBuzz with the user. The game is simple: players count in turn, starting at 1. If it is your turn, you say the next number, unless the next number is a multiple of 3 (in which case you say "Fizz"), 5 (say "Buzz") or both 3 and 5 (say "FizzBuzz"). The first player to make a mistake loses. Since the computer can't make a mistake, the program accepts one argument: a maximum number to play to. If the user makes it that far without making a mistake, the user wins. The program starts the game by saying "1".

The program should have a loop which does the following:

1. Print out the correct response (the next number, "Fizz", "Buzz", or "FizzBuzz")
2. Ask the user for the next number
3. Check to see if the user was correct. If so, continue. If not, the computer wins
4. Check to see if the next number is the stopping point. If so, the user wins. If not, go back to step 1

Note: The program must print out a message indicating who won once the game is over.

Hint: You may find it useful to create a second function that takes a number and determines what the correct output (itself, "Fizz", "Buzz", "Fizzbuzz") for it is.

[4]

```
def fizzBuzz(stop) :
    """Plays FizzBuzz until stop is reached or the user makes a mistake."""
```

MARKS

Part D: Multiple Choice [10 x 1 MARKs + 1 bonus mark]

For each of the following multiple-choice questions, circle the ***single best*** answer. The bonus question is worth 1 mark.

[1] 1. Given the following code, what is the value of `totalItems`?

```
items = [[xx,xx+1] * xx for xx in range(-1,5,2)]
totalItems = sum([len(yy) for yy in items])
```

a) 20
b) 24
c) 12
d) 10
e) 8

[1] 2. Given `word="accuratenesses"`, which of the following produces the result `"cans"`?

a) `word[-1::-3]`
b) `word[1::3]`
c) `word[2::2]`
d) `word[1:-2:2]`
e) `word[2:-1:3]`

[1] 3. Given `nn = -7 / 3`, which of the following print statements will contain **no** padding?

a) `print("{0:8.3g}".format(nn))`
b) `print("{0:8.3f}".format(nn))`
c) `print("{0:8.1E}".format(nn))`
d) `print("{0:8.6g}".format(abs(nn)))`
e) `print("{0:<8.6s}".format(str(nn)))`

[1] 4. All of the following functions exist in both the **random** library and the **numpy.random** library. Which one behaves differently in the numpy version than the normal one, assuming both were given the same inputs?

a) randint
b) randrange
c) random
d) seed
e) choice

[1] 5. Given the following code, which of the expressions below has the **largest** value?

```
w4,w2,w1,w3 = set("apple"), set("orange"), set("banana"), set("peach")
```

a) `len(w1.intersection(w2).union(w3) - w4)`
b) `len(w1.intersection(w4).union(w2) - w3)`
c) `len(w4.intersection(w3).union(w2) - w1)`
d) `len(w2.intersection(w3).union(w4) - w1)`
e) `len(w3.intersection(w1).union(w2) - w4)`

[1] 6. Once this code has been run, which of the following lines will crash?

```
import matplotlib.pyplot as plt
import numpy as np
xx = np.linspace(-10,10,101)
yy = xx**2
```

a) `plt.figure(); plt.add_subplot(321)`
b) `fig = plt.figure(); plt.clf(); fig.clf()`
c) `plt.xlim(0,10)`
d) `plt.plot(xx,yy,label="$xx$ vs $yy$")`
e) `plt.savefig("myplot.png")`

[1] 7. What is the output of the program below?

```
import numpy as np
var1 = np.ones(5)
var2 = np.linspace(1,3,5)
var2[:3:] = var1[::2]
var1[::2] += 2
var2 *= 2
print(var2 - var1)
```

a) `[-2.  0. -2.  5.  4.]`

b) `[-1.  1. -1.  4.  3.]`

c) `[-1.  1.  1.  4.  3.]`

d) `[ 0.  1.  0.  4.  4.]`

e) `[ 1. -1.  1.  2.  5.]`

[1] 8. After this code runs, how many **even** numbers have been printed?

```
counter = 0
def go(pp):
      global counter
      counter += 1
      if pp:
            print(counter)

for xx in range(5):
      go(xx)
      for yy in range(5):
            go(yy>xx)
            if counter % 4 == 0:
                  break
```

a) 1
b) 2
c) 3
d) 4
e) 5

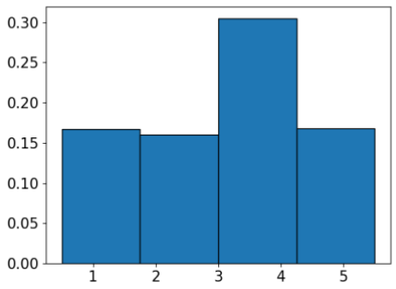
Use this space for scrap work
Questions 9 and 10 are on the next page

MARKS

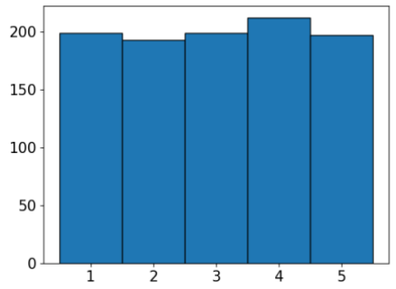
[1] 9. Given this code, which of the following plots could be produced?

```
import numpy as np
import matplotlib.pyplot as plt
nums = np.array(np.random.randint(1,6,1000))
bb = np.linspace(min(nums)-.5,max(nums)+.5,max(nums)-min(nums)+2)
plt.hist(nums,bins=bb,normed=True,ec="black")
plt.show()
```

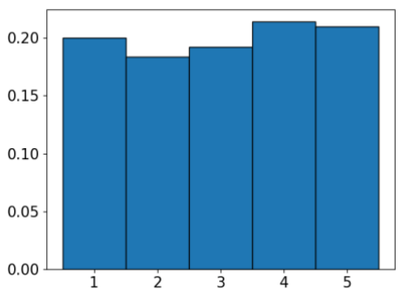
a)



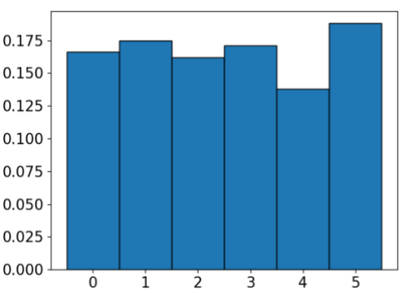
b)



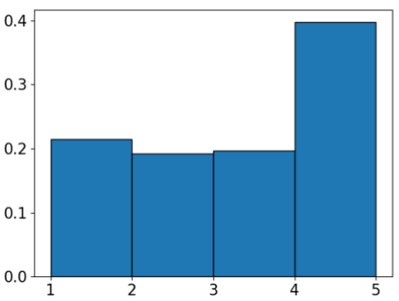
c)



d)



e)



[1] 10. Consider the following 3D array, which of the following is **false**?

```
threeD = np.reshape(np.arange(24), (4,2,3))
```

a) `np.size(threeD) = 24`
b) `np.size(threeD,axis=1) = 2`
c) `threeD[0][1][2] = 4`
d) `np.sum(threeD[1][0] < threeD[1][1]) = 3`
e) `np.shape(np.transpose(threeD)) = (3, 2, 4)`

Bonus. Recall the Alice and Bob dice game from the course notes (Rolling one die repeatedly. Alice wants every face to show up once, Bob wants one face to show up 4 times). With a 6-sided die, Bob wins about 2/3 of the time. If an 8-sided die was used instead, which of these is the closest approximation to **Bob's** winning percentage?

a) 20%
b) 40%
c) 60%
d) 80%
e) 100%

MARKS

Part E: Debugging [5 x 2 MARKS]

[10] On this page there are five small functions. Each one contains a single error. On the next page, either describe what the error is and how to fix it or rewrite the program correctly. 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).

1. This function accepts two parameters, **data** and **order**, and returns a copy of **data** sorted according to **order**. `Ex: sortBy(['a','b','c'],[3,1,2]) returns ['b','c','a']`

```
def sortOrder(data,order):
    paired = zip(order,data)
    paired.sort()
    return [item[1] for item in paired]
```

2. This function accepts two lists and returns True if **list1** is a sublist of **list2** (otherwise it returns False). This happens if and only if every item in **list1** occurs at least as often in **list2** as it does in **list1**. So [1,2,3] is a sublist of [1,2,3,4] but [1,2,2,3] is not because [1,2,3,4] only has one 2.

```
def isSublist(list1,list2):
    counts = [list1.count(item) <= list2.count(item) for item in list1]
    return sum(counts) == len(list2)
```

3. This function accepts two numpy arrays of the same length as parameters. It compares every pair of numbers (e.g. the fourth number in **array1** with the fourth number in **array2**) and returns an array containing the smaller of each pair. Ex: `getMins([1 2 3], [3 2 1]) returns [1 2 1]`

```
def getMins(array1, array2):
    mins = np.zeros(len(array1))
    mins[array1 < array2] = array1[array1 < array2]
    mins[array1 >= array2] = array1[array1 >= array2]
    return mins
```

4. This function takes two parameters: a list of items and the name of a function. It returns a list containing the results of calling **func** with each item in **args** as an argument. Ex: `applyFunction(['a','b'],str.upper) returns ['A','B']`

```
def applyFunction(args, func):
    return [func[arg] for arg in args]
```

5. This function accepts one parameter, a dictionary. This function inverts **someDict**, that is, switches its keys and values, and returns that as a new dictionary. The new dictionary's keys are the values of **someDict** and the new dictionary's values are lists (i.e. newDict[xx] is a list of all keys yy such that someDict[yy] == xx).
Ex: `invertDictionary({1:2, 3:4, 4:5, 6:5}) returns {2:[1], 4:[3], 5:[4,6]}`

```
def invertDictionary(someDict):
    newDict = {}
    for kk, vv in someDict.values():
        if vv in newDict:
            newDict[vv].append(kk)
        else:
            newDict[vv] = [kk]
    return newDict
```

MARKS

Write your corrections/rewrites below

1.

2.

3.

4.

5.