Test 1: Compsci 101

Owen Astrachan and Kristin Stephens-Martinez

February 15, 2018

Name: ANSWER Key

NetID/Login: ______________

Section Number: (01-Astrachan, 02-Stephens-Martinez): ______________

Honor code acknowledgment (signature) ________________________________

	value	grade
Problem 1	30 pts.	
Problem 2	16 pts.	
Problem 3	6 pts.	
Problem 4	8 pts.	
Problem 5	21 pts.	
TOTAL:	81 pts.	

This test has 15 pages be sure your test has them all. Do NOT spend too much time on one question — remember that this class lasts 75 minutes.

In writing code you do not need to worry about specifying the proper `import statements`. Don't worry about getting function or method names exactly right. Assume that all libraries and packages we've discussed are imported in any code you write.

Be sure your name and net-id are legible on this page and that your net-id appears at the top of every page.

There are three blank pages pages at the end of the test for extra work-space.

PROBLEM 1 : (***Name/Type/Value (30 points)***)

Part A (26 points)

Consider the following variables and their values in answering the questions below.

```
words = ["Bear", "claw", "cheesecake", "chocolate", "bar"]
say = "We don't make mistakes. We just have happy accidents."
```

Each variable in the left column is assigned a value. Provide the type and value of the variable after the assignment. The first two lines have been filled in. Types you can use are **int, float, list, string, boolean**.

Assignment	**Type**	**Value**
`a = words[0]`	String	"Bear"
`b = 5`	int	5
`c = len(words)`	int	5
`d = say[-2]`	string	“s”
`e = 5 / 2`	float	2.5
`f = 15 // 4`	int	3
`g = 6 % 4`	int	2
`h = 2 * 3 + 7`	int	13
`i = say.count("We")`	int	2
`j = say.split()[-2:]`	list	[“happy”, “accidents.”]
`k = words[2][-3] == words[3][-3]`	boolean	True
`l = words[1].upper()`	string	“CLAW”
`m = say[3:6]`	string	“don”
`n = say.split()[2]`	string	“make”
`o = say[5]`	string	“n”

Part B: Very Important Printing (4 points)

Several lines are typed into a Python console as shown below. Two of the lines printed are missing. You should indicate what is printed on these lines.

```
a = "this is a test"
w = a.split()
a2 = a
w2 = w
w.append("fun")
a = a.upper()
print(a)
THIS IS A TEST
print(a2)
            .. WHAT IS PRINTED HERE 1?
print(w)
['this', 'is', 'a', 'test', 'fun']
print(w2)
            .. WHAT IS PRINTED HERE 2?
```

What is printed as the value of `a2`

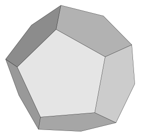
“this is a test”

What is printed as the value of `w2`

[“this”, “is”, “a”, “test”, “fun”]

PROBLEM 2 : (***Formula I (16 points)***)

Part A (8 points)



A *dodecahedron* is a twelve-sided solid as shown.

The volume of a dodecahedron is given by this formula in terms of the length of a side s:

$$\frac{15+7\sqrt{5}}{4} \times s^3$$

The surface area is given by this formula again in terms of the length of a side s.

$$s^2 \times 3 \times \sqrt{25 + 10 \times \sqrt{5}}$$

You must use the `math.sqrt` function to calculate square roots, e.g., `math.sqrt(25)` evaluates to 5. Complete both functions below to return the volume and surface areas of a dodecahedron.

```
def dodecavolume(s):
    """
    return volume of dodecahedron with side s
    """

        return (s*s*s) * (15 + 7 * math.sqrt(5))/4
```

```
def dodecasurface(s):
    """
    return surface area of dodecahedron with side s
    """

    return (s*s) * 3 * math.sqrt(25 + 10*math.sqrt(5))
```

Part B (8 points)

At Duke the cost of a *flunch* depends on the size of the group. The cost is $7.99 per person, but if the group size (including students and professors) is six or more the price is $6.50 per person. If there are two professors or more there is a 10% discount on the total cost. Write function `flunch` that has two parameters: `nums` the number of students and `nump` the number of professors. The function should return the cost of the flunch.

For example:

call	return	reason
`flunch(4,1)`	39.95	7.99×5
`flunch(3,2)`	35.955	10% off decreases 39.95 by 3.995
`flunch(5,1)`	39.00	6.50×6
`flunch(4,3)`	40.95	$6.50 \times 7 = 45.50$ and $.9 * 45.50 = 40.95$

Complete the function below:

```
def flunch(nums, nump):
    #return cost of flunch

    total = nums+nump
    cost = total*7.99
    if total >= 6:
        cost = total*6.50
    if nump >= 2:
        cost = cost * 0.90
    return cost
```

PROBLEM 3 : (***Words Are All I have (6 points)***)

Complete the body of the function `twostep` to create and return a new word by joining the first two letters of each word in a string of words separated by whitespace. When the function is completed correctly the code below will print these two lines

```
"toiled"
"futile"
```

Complete function `twostep` below. Only write code in the body of `twostep`

```
def twostep(words):
    """
    return a string based on values in parameter words
    which is a string
    """
    ret = ""
    # complete code here

    for w in words.split():
       ret = ret + w[:2]

    return ret


if __name__ == '__main__':
    ww = ["tomato illness educate",
          "funny tired lemur"]
    for w in ww:
        print(twostep(w))
```

PROBLEM 4 : (***State Farm Fences (8 points)***)

You are given two functions as shown below

```
def onepost(n):
    s = r"|----"
    return s*n + "|"

def twoposts(n):
    s = r"||----"
    return s*n+"||"
```

To see what the functions return consider the four calls below:

```
    print(onepost(5))
    print(onepost(5))
    print(twoposts(3))
    print(twoposts(3))
```

These calls generate this output

```
|----|----|----|----|----|
|----|----|----|----|----|
||----||----||----||
||----||----||----||
```

You must create and write a function `makefence` so that the function calls below generate the output that follows. Note that the first parameter to `makefence` is the width of the fence (related to the number of posts) and the second parameter is the height of the fence, the number of lines printed. The third parameter will be one of `onepost` or `twoposts` as shown, it's the name of the function. Write the complete function on the next page, including the parameters.

```
    makefence(5,3,onepost)
    makefence(6,2,twoposts)
```

```
|----|----|----|----|----|
|----|----|----|----|----|
|----|----|----|----|----|
||----||----||----||----||----||----||
||----||----||----||----||----||----||
```

Complete the function below. Be sure to specify the names and order of the parameters in the space given. The function prints, it does not return a value.

```
def makefence(   width, height, func                        )
    """
    first parameter is an int, the number of cross-pieces
    second parameter is an int, the height of the fence
    third parameter is a function to call that returns
    a string representing the cross-pieces, e.g., twoposts
    print a fence according to specifications
    """
     for _ in range(height):
         print(func(width))


           OR


      str = func(width)
      fence = str + “\n”
      all = fence*height
      print(all.strip())
```

PROBLEM 5 : (*Cement Incorporated (21 points)*)

In this problem you're given a text file of data and code that reads the file so that the data can be processed by functions you write.

At a cement company orders come in based on the day the cement is needed, the amount of cement in cubic yards, and the grade of cement (M10, M20, or M30). Orders are kept in a file in the following format: `MM/DD/YYYY:AMOUNT:GRADE`. So for example in the file `"cement.txt"` below the first line represents an order on December 24, 2012 for 12 cubic yards of grade M30 cement. The last line is for an order on June 22, 2016 for 1 cubic yard of grade M20 cement.

```
12/24/2012:12:M30
10/17/2016:12:M30
4/29/2013:10:M10
6/5/2012:11:M20
6/19/2017:10:M30
7/2/2011:6:M30
4/8/2012:9:M30
2/20/2017:19:M30
2/26/2015:11:M20
10/28/2016:20:M20
9/9/2017:12:M30
5/24/2013:5:M30
6/26/2012:3:M20
10/31/2013:6:M30
12/10/2010:15:M20
6/22/2016:1:M20
```

A function `fileToList` reads a datafile in the format above and returns a list of lists in the format shown below. Each list that is an element of the list returned represents one line of the data file and has three values: the date as a string, the amount of cement as an int, and the grade as a string.

```
>>> datalist = fileToList("cement.txt")
>>> datalist
[['12/24/2012', 12, 'M30'],
 ['10/17/2016', 12, 'M30'],
 ['4/29/2013', 10, 'M10'],
    ... NOT ALL SHOWN ...
 ['6/22/2016', 1, 'M20']]
```

(problems continued)

Part A (8 pts)

Complete function `fileToList` below. As described above, the function has one parameter, `filename`, a string that is the name of the file in the specified input format. This function reads in the file and returns a list of three-element lists in the format specified earlier, where each three element list represents a single cement order. As shown above, each three element list contains a string, and int, and a string.

Function `fileToList` is started below. Complete the function so that it works as specified.

```
def fileToList(filename):
    """
    Read the data in filename (string). Return a list of lists
    where each inner list of the list returned
    is three elements: [string, int, string]  representing one line
    of the file read
    """
    result = []
    f = open(filename)
    for line in f:
        line = line.strip()

         date = line[0]
         amt = int(line[1])
         grade = line[2]
         result.append([date,amt, grade])

    f.close()
    return result
```

Part B (6 pts)

Implement the function `cementForDate` that has one list parameter, a list of three-element lists as described above, and a second string parameter representing a date. The function `cementForDate` returns the total number of cubic yards ordered on the given day. For example:

```
>>> datalist = fileToList("smalldata.txt")
>>> datalist
[['12/24/2012', 12, 'M20'],
['10/17/2016', 12, 'M20'],
['12/24/2012', 10, 'M10']]
>>> cementForDate(datalist, '10/17/2016')
12
>>> cementForDate(datalist, '12/24/2012')
22
>>> cementForDate(datalist, '12/24/2017')
0

def cementForDate(data, day):
    """
    Returns the total cement in data (list) ordered on day (string).
    """

    total = 0
    for one in data:
        if one[0] == day:
            total = total + one[1]

    return total
```

Part C (3 pts)

A cement truck can only hold nine cubic yards. The function `trucksNeeded` is supposed to return the number of trucks needed for the number of yards specified by parameter `yards`. The function does not work as intended. In the output below, the value returned by the call `trucksNeeded(9)` should be 1, but the value 2 is returned.

For example:

```
>>> trucksNeeded(8)
1
>>> trucksNeeded(9)
2
>>> trucksNeeded(10)
2
```

Fix the function `trucksNeeded` so it works as intended for all non-negative values of parameter `yards`.

```
def trucksNeeded(yards):
    """
    Return the number of cement trucks needed for the number of cubic yards
    specified by int parameter yards.
    """
```

~~return yards//9 + 1~~

```
    if yards % 9 == 0:
        return yards//9

    return yards//9 + 1
```

Part D (4 pts)

Write the function `trucksForDate` that returns the number of trucks needed for all the orders in parameter `data` that occur on the date specified by parameter `day`. The parameter `data` is a list of three-element lists as described above and day is a string representing a date.

In writing `trucksForDate` you may call functions `cementForDate` and `trucksNeeded` that were specified above. Assume these functions work correctly, as specified. Do not duplicate the functionality provided by these functions.

Using the textfile `"cement.txt"` file shown at the beginning of this problem to create `datalist` the call `trucksForDate(datalist, "12/10/2010")` should return 2 since 15 cubic yards are needed on that date so 2 trucks are needed.

```
def trucksForDate(data, day)
    """
    Returns the total number of trucks needed in data (list) for
    a given day (string).
    """

    yards = cementForDate(data,day)
    return trucksForDate(yards)
```

extra page

extrapage