Test 1: Compsci 101

Kristin Stephens-Martinez

October 14, 2019

Name: _____ (1/2 pt)

NetID/Login: _____ (1/2 pt)

Honor code acknowledgment (signature) _____ (1 pt)

	Value	Grade
Front Page	2 pts.	
Problem 1	28 pts.	
Problem 2	9 pts.	
Problem 3	12 pts.	
Problem 4	8 pts.	
Problem 5	6 pts.	
Problem 6	6 pts.	
Problem 7	10 pts.	
TOTAL:	81 pts.	

Bubble sheet: Tear off the bubble sheet from the front of the exam and fill in the Name, ID, and Date field. Your ID is your NetID.

This test has 14 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 NetID are legible on this page and that your NetID appears at the top of every page.

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

PROBLEM 1 : (*Name/Type/Value (28 points)*)

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

```
states = ['Kansas', 'West Virginia', 'Georgia', 'Michigan', 'New Jersey',
          'Wisconsin', 'Colorado']
say = 'Old McDonald had a farm. Ee-igh, ee-igh, oh!'
```

For each statement, provide the type and value after the statement is executed. For example, for the code `5` the type is `int` and the value is `5`. Or for the code `states[2]` the type is `string` and the value is `'Georgia'`

`say[-3:]`

1. What is the type? (a) boolean (b) float (c) int (d) list (e) string
2. What is the value? (a) `'oh!'` (b) `'h!'` (c) `'o'` (d) `' '` (e) `' oh!'`

`states[-2]`

3. What is the type? (a) boolean (b) float (c) int (d) list (e) string
4. What is the value? (a) `'Georgia'` (b) `'Wisconsin'` (c) `'Colorado'` (d) `'New Jersey'`

`(3 < 9) and (7 == 7)`

5. What is the type? (a) boolean (b) float (c) int (d) error (e) none
6. What is the value? (a) `True` (b) `False` (c) `Error` (d) `None`

`'Michigan' in states`

7. What is the type? (a) boolean (b) float (c) int (d) error (e) none
8. What is the value? (a) `True` (b) `False` (c) `Error` (d) `None`

`11.0 + 4.0`

9. What is the type? (a) boolean (b) float (c) int (d) list (e) string
10. What is the value? (a) `15.0` (b) `7.0` (c) `14` (d) `15` (e) `14.0`

`9 % 3`

11. What is the type? (a) boolean (b) float (c) int (d) list (e) string
12. What is the value? (a) `1` (b) `0` (c) `2` (d) `3`

`len(say.split('d'))`

13. What is the type? (a) boolean (b) float (c) int (d) list (e) string
14. What is the value? (a) `4` (b) `5` (c) `3`

```
99 / 11
```

15. What is the type? (a) boolean (b) float (c) int (d) list (e) string

16. What is the value? (a) `9` (b) `11` (c) `9.9` (d) `11.0` (e) `9.0`

```
'y' in states
```

17. What is the type? (a) boolean (b) float (c) int (d) error (e) none

18. What is the value? (a) `True` (b) `False` (c) `Error` (d) `None`

```
states[1:3]
```

19. What is the type? (a) boolean (b) float (c) int (d) list (e) string

20. What is the value?
    (a) `['West Virginia', 'Georgia']`
    (b) `['West Virginia', 'Georgia', 'Michigan']`
    (c) `['Georgia', 'Michigan']`
    (d) `['Kansas', 'West Virginia', 'Georgia']`
    (e) `['Kansas', 'West Virginia']`

```
states[3][2]
```

21. What is the type? (a) boolean (b) float (c) int (d) list (e) string

22. What is the value? (a) `'o'` (b) `'w'` (c) `'h'` (d) `'i'` (e) `'c'`

```
8 // 3
```

23. What is the type? (a) boolean (b) float (c) int (d) list (e) string

24. What is the value? (a) `2.6666666666666665` (b) `2` (c) `4` (d) `1` (e) `3`

```
',' in say
```

25. What is the type? (a) boolean (b) float (c) int (d) error (e) none

26. What is the value? (a) `True` (b) `False` (c) `Error` (d) `None`

```
'old' in say
```

27. What is the type? (a) boolean (b) float (c) int (d) error (e) none

28. What is the value? (a) `True` (b) `False` (c) `Error` (d) `None`

PROBLEM 2 : (*What will Python display? (9 points)*)

Given the code blow, choose the return value for each of the function calls.

```
def mystery(lstNum):
    lstNum1 = []
    lstNum2 = []

    for elem in lstNum:
        num1 = elem[0]
        num2 = elem[1]

        if num1 not in lstNum1:
            lstNum1.append(num1)
            lstNum2.append(0)

        currIdx = lstNum1.index(num1)
        lstNum2[currIdx] += num2

    retLst = []
    for i in range(len(lstNum1)):
        retLst.append([lstNum1[i],lstNum2[i]])

    return retLst
```

```
mystery([[1, 1], [1, 2], [2, 2]])
```

29. What is the value?

(a) `[1, 3, 2, 2]`
(b) `[[1, 3], [2, 2]]`
(c) `[[1, 2], [2, 1]]`
(d) `[[1, 3], [2, 3]]`

```
mystery([[3, 3], [2, 2], [1, 1]])
```

30. What is the value?

(a) `[[3, 2], [2, 2], [1, 2]]`
(b) `[[1, 1], [2, 2], [3, 3]]`
(c) `[[3, 3], [2, 2], [1, 1]]`
(d) `[[3, 1], [2, 1], [1, 1]]`
(e) `[3, 3, 2, 2, 1, 1]`

```
mystery([[4, 4], [2, 2], [2, 4]])
```

31. What is the value?

(a) `[[2, 6], [4, 4]]`
(b) `[[4, 4], [2, 6]]`
(c) `[[4, 1], [2, 2]]`
(d) `[4, 4, 2, 6]`
(e) `[[2, 2], [4, 1]]`

PROBLEM 3 : (*Spot the Bug (12 points)*)

The following function removes all occurrences in the given string of the given character, except if that character appears as the first or last letter. But this function is buggy! After answering parts A and B about this code, you will fix it in part C. Below are some example calls of what the function should return.

Function call	**Correct return value if there was no bug**
`removeInnerLetters('please', 'l')`	`'pease'`
`removeInnerLetters('apple', 'a')`	`'apple'`
`removeInnerLetters('rat', 't')`	`'rat'`
`removeInnerLetters('dodged', 'd')`	`'doged'`
`removeInnerLetters('ball', 'l')`	`'bal'`
`removeInnerLetters('to', 's')`	`'to'`
`removeInnerLetters('a', 'a')`	`'a'`
`removeInnerLetters('a', 't')`	`'a'`

```
def removeInnerLetters(word, letter):
    newWord = word[0]
    for c in word[1:-1]:
        if c != letter:
            newWord += c
    return newWord + word[-1]
```

Part A (4 points) The following two questions are like checkbox questions, a.k.a. multiple options could be filled in on your answer sheet.

32. Of the following function calls, fill in which calls the buggy code *WILL NOT* work as expected.
    (a) `removeInnerLetters('please', 'l')` should return `'pease'`
    (b) `removeInnerLetters('apple', 'a')` should return `'apple'`
    (c) `removeInnerLetters('rat', 't')` should return `'rat'`
    (d) `removeInnerLetters('dodged', 'd')` should return `'doged'`
    (e) None of the above

33. Of the following function calls, fill in which calls the buggy code *WILL NOT* work as expected.
    (a) `removeInnerLetters('ball', 'l')` should return `'bal'`
    (b) `removeInnerLetters('to', 's')` should return `'to'`
    (c) `removeInnerLetters('a', 'a')` should return `'a'`
    (d) `removeInnerLetters('a', 't')` should return `'a'`
    (e) None of the above

Part B (4 points)

In the first two cells below, provide the arguments for a call to `removeInnerLetters` with either your own arguments or arguments from the examples that returns a *wrong* value. In the "actual return value" cell, write your function call's return value. If the function call causes an error, write "Error" in the cell. In the "correct return value" cell, write the value it should return.

argument: `word`	argument: `letter`	actual return value	correct return value

Part C (4 points)

Here is the buggy code again. Fix it so that it always returns the correct values.

```
def removeInnerLetters(word, letter):


    newWord = word[0]


    for c in word[1:-1]:


        if c != letter:


            newWord += c


    return newWord + word[-1]
```

PROBLEM 4 : (*Formulas (8 points)*)

Part A (4 points)

$F = G\frac{m_1 m_2}{r^2}$ $G = 6.674 \times 10^{-11}$.

Newton's law of universal gravitation equation of the force attracting two particles is given above. m_1 and m_2 are the mass of each particle, r is the distance between them, and G is the gravitational constant.

Implement the function `force` that takes the mass of two particles and their distance and returns the force acting between the two particles.

```
def force(mass1, mass2, distance):
    '''
    mass1 (int or float) - the mass of the first point under consideration
    mass2 (int or float) - the mass of the second point under consideration
    distance (int or float) - the distance between the two points

    Return the force attracting MASS1 to MASS2 when they are DISTANCE away from
    each other. Assume the gravitational constant G is 6.674*10^-11.
    '''
```

Part B (4 points)

$A = P(1 + \frac{r}{n})^{nt}$

The compound interest formula is above. A is the future value of the investment. P is the principal or original amount of the investment. r is the yearly interest rate (e.g. 5% makes $r = 0.05$). n is the number of times the interest will be compounded within a year. t is the number of years the money is invested.

Implement the function `furtureValue` that uses the formula to return the future value of the investment.

```
def futureValue(principle, interest, num, time):
    '''
    principle (int or float) - the initial amount of the investment
    interest (float) - the yearly interest rate on the principle
    num (int) - the number of times the interest will be compounded per year
    time (int) - the amount of time the money is invested in years

    Return the future value of an investment based on the PRINCIPAL,
    INTEREST, NUM, and TIME.
    '''
```

PROBLEM 5 : (*Divisor counting (6 points)*)

Implement the function `countDivisors` as described below. Assume it is called with positive integers greater than 0.

Example calls are:

Function call	**Return value**	**Divisors**
`countDivisors(4)`	`3`	1, 2, and 4
`countDivisors(1)`	`1`	1
`countDivisors(7)`	`2`	1 and 7
`countDivisors(10)`	`4`	1, 2, 5, and 10

```
def countDivisors(n):
    '''
    n (int) : A positive integer greater than 0

    Return the number of positive divisors for the value N. A divisor for N is
    an integer that evenly divides into N (i.e. the remainder is 0). Divisors
    for N also include 1 and N itself.
    '''
```

PROBLEM 6 : (*Zip it up (6 points)*)

Implement the function `zipper` as described below.

Example calls are:

Function call	**Return value**
`zipper([1, 2, 3], [5, 7, 2], max)`	`[5, 7, 3]`
`zipper([7, 2, 5], [9, 3, 6], min)`	`[7, 2, 5]`
`zipper([2, 3, 4], [2, 2, 2], pow)`	`[4, 9, 16]` # pow(x,y) means x^y

```
def zipper(lst1, lst2, merge):
    '''
    lst1 (list) - a list of elements
    lst2 (list) - a list the same length as lst1
    merge (function) - a function that takes two parameters of the same types
        as lst1 and lst2

    Return a new list that is the result of calling MERGE on each element of
    LST1 and LST2. The ith element of the returned list should be the return
    value of calling MERGE with the ith element of LST1 as the first argument
    and the ith element of LST2 as the second argument.
    '''
```

PROBLEM 7 : (*Hopping Lists (10 points)*)

Write a function called `crazyHopping` that takes a list of integers greater than or equal to 0. Then, starting at index 0, it will "hop" to the element specified by the current integer element by using its value as the next index. The code will continue to do this until either it reaches an index value that is outside of the list or it has made more hops than there are elements in the list and therefore is in some infinite loop. If the former happens, the function will return the number of hops it made to escape the list. If the latter happens, the function will return -1.

Function Call	**Return Value**	**Explanation**
crazyHopping([2, 3, 1])	3	The function starts at index 0, then goes to index 2, then index 1, and finally reaches the integer 3, which is outside of the list. This was a total of 3 hops to escape the list.
crazyHopping([1, 0])	-1	The function starts at index 0, goes to index 1, then back to index 0, and finally back to index 1. At this point, it's now made 3 hops, which is longer than the length of the list and has not reach a number to escape the list, therefore it returns -1.
crazyHopping([2, 5, 5, 5])	2	The function starts at index 0, goes to index 2, and finds a 5, which leads to outside the list. This was a total of 2 hops to escape the list.
crazyHopping([1])	1	The function starts at index 0, which is the value 1 that is outside of the list. Therefore, it took 1 hop to escape the list.

```
def crazyHopping(lst):
    """
    lst (list of ints) - List of integers >= 0

    Returns how many hops it takes to escape the list or -1 if the list cannot
    be escaped.
    """
```

PROBLEM 8 : (*Extra Credit (1 point)*)

Predict what range your percentage grade will be on the exam. If you are correct, you will earn 1 point of extra credit on this exam, rounding in your favor (e.g. you will earn a point if your score is 94.5% and you choose the 90%-94% range). The front page has a table with the number of points for each problem.

- ◯ 95% - 100%
- ◯ 90% - 94%
- ◯ 85% - 89%
- ◯ 80% - 84%
- ◯ 75% - 79%
- ◯ 70% - 74%
- ◯ 65% - 69%
- ◯ 60% - 64%
- ◯ 55% - 59%
- ◯ 50% - 54%
- ◯ 45% - 49%
- ◯ 40% - 44%
- ◯ 35% - 39%
- ◯ 30% - 34%
- ◯ 25% - 29%
- ◯ 20% - 24%
- ◯ 15% - 19%
- ◯ 10% - 14%
- ◯ 5% - 9%
- ◯ 0% - 4%

extra page

extra page

extrapage