Computer Science 1 — CSci 1100 — Fall 2022
Final Exam
December 14, 2022

SOLUTIONS

1. (**12 points**) Show the output of the following:

Code:

```python
# Part (a)
creatures = {'whale': 0, 'ostrich': 2, 'spider': 8, \
             'crocodile': 4, 'robin': 2, 'tiger': 4, \
             'caterpillar': 16, 'chipmunk': 4}
print(list(filter(lambda anim: \
                  2 <= creatures[anim] < 8, \
                  creatures.keys())))
by_legs = list(map(lambda legs: ( \
    legs, [anim for anim in creatures.keys() \
           if creatures[anim] == legs]), \
                          set(creatures.values())))
print(sorted(by_legs, key = lambda el: len(el[1]), \
             reverse = True)[0])
print("\n".join(["{}:\t{}". \
                 format(el[0], '*' * len(el[1])) \
                 for el in sorted(by_legs)]))
```

Solution:

```
['ostrich', 'crocodile', 'robin', \
                'tiger', 'chipmunk']
(4, ['crocodile', 'tiger', 'chipmunk'])
0:  *
2:  **
4:  ***
8:  *
16: *
```

Code:

```python
# Part (b)
def is_special(word):
  if len(word) < 2:
    return True
  return word[0]==word[-1] and is_special(word[1:-1])
print(is_special("Radar"))
print(is_special("racecar"))
print(is_special(""))
print(is_special("lemma"))
print(is_special("I"))
```

Solution:

```
False
True
True
False
True
```

2. (**12 points**) For each of the following write a **single line of Python code** to solve the problem. You can assume you are typing the command into the Python Shell. You may use any combination of the techniques we have discussed throughout the semester, including list comprehensions, map, filter and lambda functions. However, any use of a `for` loop or an `if` (except within a list comprehension), or a `while` loop will receive a 0. Solutions of more than 1 line will not receive full credit.

Code:	**Solution:**
(a) Given a list of strings `L1`, write code that evaluates to a list with the even length strings removed and with the odd length strings reversed. For example, if `L1 = ["garter", "cobra", "boa", "corn"]` the code should output `['arboc', 'aob']`	`[x[::-1] for x in L1 if len(x)%2 == 1]` or `list(map(lambda x:x[::-1],filter(lambda x: len(x)%2 == 1, L1)))`
(b) Given a list of 3-Tuples, generate a new list sorted first by the second value of the tuple, and then by the sum of the remaining two values. So given: `L1 = [(5, 1, 2),(2, 0, 2),(4, 1, 1),(1, 0, 1)]` Your code should generate: `[(1, 0, 1), (2, 0, 2), (4, 1, 1), (5, 1, 2)]`	`sorted(L1, key=(lambda x: (x[1], x[0]+x[2])))` or `sorted(sorted(L1, key=(lambda x: x[0]+x[2])), \ key=(lambda x: x[1]))`
(c) Given a dictionary `D` with keys and values both lowercase strings, write code that evaluates to the total number of `'e'`s in the keys omitting all keys that start with `'e'`. So, as an example, if: `D = {"eric": "idle", "john": "cleese", "terry": "gilliam"}` then the output should be 1. (The keys are `{'eric', 'john', 'terry'}`). We do not count eric because it starts with `'e'` and there is exactly one `'e'` in `{'john', 'terry'}`). You may assume all keys are at least one character long.	`sum([x.count('e') for x in D if x[0] != 'e'])` or `sum(map(lambda x: x.count('e'), \ filter(lambda x: x[0] != 'e', D)))`

3. (**10 points**) Consider the binary search algorithm discussed in class below:

```
def binary_search( x, L):
    low = 0
    high = len(L)
    while low != high:
        mid = (low+high)//2
        if x > L[mid]:
            low = mid+1
        else:
            high = mid
    return low
```

You are given a list of English characters called `letters` that is sorted in non-decreasing order, and a character `target`. There are at least two different characters in the list `letters`. Write a function `next_letter(letter,target)` that uses the binary search algorithm to find the smallest character in `letters` that is lexicographically greater than the `target`. If such a character does not exist, return the first character in `letters`. Hint: You may modify the binary search to return the desired result but the order complexity `O(logn)` should not be altered. Example runs:

```
>>>letters = ["c","f","j"]
>>> target = "a"
>>> print(next_letter(letters,target))
c
>>>letters = ["c","f","j"]
>>> target = "c"
>>> print(next_letter(letters,target))
f
>>>letters = ["x","x","y","y"]
>>> target = "z"
>>> print(next_letter(letters,target))
x
>>>letters=["a","f","x"]
>>>target="x"
>>>print(next_letter(letters, target))
a
```

Solution:

```
def next_letter(self, letters, target):
        low,high=0,len(letters)-1
        while low <= high:
            mid=low+(high-low)//2
            if letters[mid]<=target:
                low=mid+1
            else:
                high=mid-1
        return letters[low%len(letters)]
```

4. (**16 points**) Among the many sorts we didn't cover is the radix sort. The code below is an implementation of radix sort for numbers up to two digits. The idea of the sort is pretty simple. Start by making 10 piles corresponding to the digits `0 - 9` (lines #1 - #3). Now go through the elements of the list and looking only at the ones digit, put the elements in the correct pile (lines #4 - #7). Next pull the piles together into the list again, and empty the piles (lines #8 - #11). Now go onto the tens digit and do it all over again. At the end, the list L is sorted.

```
def radixsort(L):
    Ls = []                         #1
    for index in range(10):         #2
        Ls.append([])               #3
    pos = 0                         #4
    for el in L:                    #5
        index = get_digit(el, pos)  #6
        Ls[index].append(el)        #7
    L.clear()                       #8
    for el in Ls:                   #9
        L.extend(el)                #10
        el.clear()                  #11
    pos = 1                         #12
    for el in L:                    #13
        index = get_digit(el, pos)  #14
        Ls[index].append(el)        #15
    L.clear()                       #16
    for el in Ls:                   #17
        L.extend(el)                #18
        el.clear()                  #19
```

(a) (10/16) Write the function `get_digit(value, pos)` used above. Given an integer `value`, `get_digit` returns the digit at position `pos` of the integer. Note that position 0 is the ones digit. If `pos` asks for a digit not in value, simply return 0. For example, we have:

```
>>> get_digit(12, 0)
2
>>> get_digit(12, 1)
1
>>> get_digit(12, 2)
0
```

Solution:

```
def get_digit(val, pos):
    val = str(val)[::-1]
    if pos >= len(val):
        return 0
    return int(val[pos])
```

or

```
def get_digit2(val, pos):
    div = 10**pos
    return (val // div) % 10
```

(b) (6/16) Our radix sort only works for integers of up to 2 digits, but it is a simple extension to make it work for arbitrary length integers. Write a new version `radixsort(L, n)` that takes a list `L` and an integer `n`. The new radix sort must sort integers of up to `n` digits. **Solution:**

```python
def radixsort2(L, n):
    Ls = []
    for i in range(10):
        Ls.append([])
    pos = 0
    while pos < n:
        for l in L:
            index = get_digit2(l, pos)
            Ls[index].append(l)
        L.clear()
        for l in Ls:
            L.extend(l)
            l.clear()
        pos += 1
```

5. (**16 points**) Consider the mathematical formula $a_n = 2a_{n-1} - a_{n-2} + 2$ with $a_1 = 14$, and $a_2 = 17$.

(a) (12/16 points) Write a recursive function, `a`, that takes one integer parameter, `n`, and returns the value of the function defined by the formula.

Solution:

```
def a(n):
    if n == 1:
        return 14
    if n == 2:
        return 17
    return 2*a(n-1) - a(n-2) + 2
```

(b) (2/16 points) What are the base cases for the function `a`?

Solution:

```
n = 1 and n = 2
```

(c) (2/16 points) For what integer values of `n` is the function undefined?

Solution:

```
n < 1
```

6. (**14 points**) Think back to the attributes list we used for our houses descriptions. The attributes list was a list of dictionaries where each dictionary contained entries for features of the house such as a list of offers,

   Assume you have three 2-tuples, d1, d2, and d3. For example, they could be:

```
d1 = ("id", {12: 311234, 16: 392657, 18: 199762, 20: 203461})
d2 = ("shutters", {311234: "blue", 2033461: "grey"})
d3 = ("offers", {311234: [300, 150], 199762: [190], 203461: [120, 150, 180]})
```

   although you cannot assume these exact values.

   For each tuple, the first element of the tuple is the name of an attribute and the second is a dictionary of values for that attribute for a number of houses. You can assume that d1 has the id attribute. The dictionary for d1 has house numbers as keys and identifiers as values. The dictionary for d2 and d3 have identifiers as keys and the values correspond to the attribute of the list. For the examples shown, for d2 the values are the colors of the shutters on the different houses and for d3 the values are lists of offers in thousands of dollars.

   Assume that d1, d2 and d3 are already defined. Write code to create an attribute list called houses where each entry in the list is a dictionary for one of the houses with the attributes as keys and the attribute values as values. For example, for d1, d2 and d3 as above, you should have:

```
>>> print(houses)
[{'id': 311234, 'shutters': 'blue', 'offers': [300, 150]}, \
 {'id': 392657}, \
 {'id': 199762, 'offers': [190]}, \
 {'id': 203461, 'offers': [120, 150, 180]}]
```

   Be careful. Not all identifiers appear in d2 or d3.

   Solution:

```
houses = []
for num in d1[1]:
    ident = d1[1][num]
    house = {d1[0]: d1[1][num]}
    if ident in d2[1]:
        house[d2[0]] = d2[1][ident]
    if ident in d3[1]:
        house[d3[0]] = d3[1][ident].copy()
    houses.append(house)
```

7. (**16 points**) Define a class called tictactoe that has an initialization method, a mark method and a `str` method. The class maintains a `3x3` grid. The initialization class should create an empty grid. The mark method should take a symbol, a row, and a column and place the symbol in the grid at the position requested unless there is already something there. If there is a symbol already there it should do nothing. You can use a space to indicate an empty location. The `str` method should take the data and return the rows of the grid with the `|` symbol separating each column and a row of `'-----'` between each row. For example:

```
>>> t = tictactoe()
>>> print(t)
 | |
-----
 | |
-----
 | |
>>> t.mark('O', 1, 1)
>>> print(t)
 | |
-----
 |O|
-----
 | |
>>> t.mark('X', 2, 1)
>>> print(t)
 | |
-----
 |O|
-----
 |X|
```

Write your answer on the next page.

Solution:

```
class tictactoe(object):
    def __init__(self):
        self.board = []
        self.board.append(['', '', ''])
        self.board.append(['', '', ''])
        self.board.append(['', '', ''])

    def mark(self, symbol, row, col):
        if self.board[row][col] == '':
            self.board[row][col] = symbol

    def __str__(self):
        string = ''
        for row in self.board:
            for symbol in row:
                if symbol == '':
                        symbol = ' '
                    string = string + symbol + '|'
            string = string.strip('|')
            string += '\n-----\n'
        string = string.strip('-\n')
        return string
```

8. (**2 points**) Please read carefully and respond below:

   (a) All grades except Lab 11, Lab 12, HW 8 and the final exam are frozen and unchangeable. Our syllabus specifies 7 days for grade inquiries and grades have been released for longer than that on all of the previous gradeables. Note the grade inquiries that have already been reported will still be attended to.

   (b) Grades for Lab 11, Lab 12, and HW 8 will be frozen and unchangeable after Thursday, 12/15 at 5 pm EDT. Before that time you must check your Rainbow grades for correctness and appeal any issues with Lab 11 or 12 grades (to your lab TA) and HW 8 grading (to the TA who graded).

   (c) Our intended grading schedule is as follows. Please note that our TAs have their own final exams. We will do our best to make this schedule, but if we run into issues, we may need to slip. This may affect the amount of time we can give you to request regrades. **Any changes will be posted to the Discussion Forum on Submitty. We will make sure that you have at least 12 hours to request regrades.**:

      i. Sometime before Saturday 12/17 at 11:59 pm grades will be released for the final exam. These grades will have already been checked carefully. You have until Sunday 12/18 at 11:59 pm to request regrades. After that time all final exam grades will be be frozen.

      ii. Final grades for the course will be completed and posted by 5pm Monday 12/19.

   (d) Please write **True** (2 points) below to indicate that you have read and understood these instructions.

```
True
```

Congratulations! CS-1 is over. Go forth, have some fun, and enjoy the break!
Remember: We are always looking for good mentors! Please look for Shianne Hulbert's note if you are interested, or contact her via email!

This part is completely optional and will not affect your grade: Use the space below to draw art work to entertain us (and make you famous) during final exam grading.