MTH 3300
E. Fink

3300 Problems, Section 8: Dictionaries; pandas & matplotlib; Recursion; Searching and Sorting

1. I have a dictionary defined by

   ```
   the_dict = {"a":1, "b":3, "c":5, "d":2, "e":6}
   ```

   Write code that adds the key `"f"` to this dictionary with value `0`. Then, write code (using a loop) which prints each key-value pair in `the_dict` on its own line.

2. I have a dictionary defined by

   ```
   new_dict = {"a":1, "b":3, "c":5, "d":2, "e":6}
   ```

   Write code that asks the user to input a letter, and then either adds 1 to the appropriate value (if the input is already a key in `new_dict`), or creates a new entry in `new_dict` (if the input is not already a key).

3. Suppose that I have the following code:

   ```
   n = ["Joey", "Frank", "Lenny", "Mary", "Caroline", "Tony"]
   values = [40, 80, 225, 60, 90, 55]
   ```

   Write code that will combine these lists into a dictionary named `d`, using a loop (or a zip, if you prefer). The names should be the keys, so that afterwards, for example,

   ```
   print(d["Joey"] + d["Frank"])
   ```

   would cause `120` to print on the console.

4. Write a function called `find_value`, which takes a dictionary `dic` and a value `val` as arguments. The function should return a `list` of all the keys whose value in `dic` is equal to `val`.

5. Suppose that I have a sentence contained in a long string variable named `x`. For example

   ```
   x = "this is a long sentence it is a sentence with words"
   ```

   Create a dictionary called `counts`. The keys in `counts` will be all the words that appear in `x`; the values will be the number of times that each word appears. You may assume no punctuation, all the characters are lowercase, and there is a space between each word in the string.

6. I have two lists of strings named `fruits` and `prices`. The first is a list of fruits for sale at a store; the second is a list of how much each fruit costs at my store (in the corresponding order). For example, we could have

   ```
   fruits = ["Pear", "Lime", "Banana"]
   ```

   and

   ```
   prices = [1.25, 0.85, 1.50]
   ```

   (where here a pear costs $1.25, a lime costs $0.85, and a banana costs $1.50).

   a. Write code which creates a dictionary called `fp`. This dictionary should contain the entries of `fruits` as keys, with the corresponding entries of `prices` as values. **Your code should work with any two lists of strings named `fruits` and `prices` which are the same length.**

   b. Write code that allows the user to enter the names of three fruits, and then prints out the total cost of purchasing those fruits – *making sure to check that the fruit they request is being sold!* If the user requests a fruit which is not sold, simply ignore that item. **FOR FULL CREDIT, use `fp`, and do NOT use `fruits` and `prices`.**

7. I have two lists of strings named `children` and `parents`. The first is a list of children; the second is a list of the corresponding parents who are supposed to pick them up from school (in the corresponding order). For example, we could have

   ```
   children = ["Bobby", "Susie", "Johnny"]
   ```

   and

   ```
   parents = ["Mr. Johnson", "Ms. Smith", "Mr. Williams"]
   ```

(where `Mr. Johnson` is `Bobby`'s father, etc.).

a. Write code which creates a dictionary called `pickup`. This dictionary should contain the entries of `children` as *keys*, with the corresponding entries of `parents` as *values*. **Your code should work with any two lists of strings named `children` and `parents` which are the same length.**

b. I have a list called `present`, containing PARENTS who are currently here for pickup. By **looping through the `pickup` dictionary (for full credit!)**, create a list of CHILDREN whose parents are NOT in `present`. (Hint: to make this shorter, remember that you can check the presence of an element `x` in a list `y` by using the code `if x in y`.)

8. Suppose that I have a CSV spreadsheet file named `sales.csv`. The spreadsheet contains the following data, with the given column names.

	customer	sale	date
0	Albert	238.99	01/01/2014
1	John	162.54	01/02/2014
2	John	201.82	01/05/2014
3	Albert	200.40	01/08/2014
4	Albert	57.52	01/10/2014
5	Peter	196.54	01/21/2014

a. Create a pandas DataFrame named `x` from the file, which contains this data. (One line.)

b. Create a new, smaller DataFrame called `y`, which contains only the rows where `Albert` is the customer, and only the columns `customer` and `sale`.

c. By looping through `x` (or some other pandas technique, if you like), write code which computes the sum of the values of the `sale`s in the entire (original) table.

9. Suppose that I have a CSV spreadsheet file named `shark.csv`. The spreadsheet contains the following data:

	month	shark_attacks	ice_cream_sales
0	May	10	13201
1	June	3	5706
2	July	14	18115
3	August	19	23648

a. Write code which creates a pandas DataFrame, containing this data. Then write code which prints out the last two columns of this DataFrame. (It doesn't have to be printed in any particularly nice manner, and don't worry about Python omitting some rows in its display.)

b. Now, print only the rows that have ≥ 10 shark attacks.

10. Consider the following list of data, which represents how much of a computer's CPU is being utilized at time t seconds, for $t = 0$ to $t = 8$:

```
percents = [0.52, 0.56, 0.61, 0.60, 0.61, 0.57, 0.49, 0.51, 0.38]
```

Write code that uses `matplotlib` to create a line graph showing the utilization versus time. Label the axes as you see fit, and give the chart a title.

11. I have a CSV spreadsheet contained in a file named `grants.csv`, which contains the following data (the top line contains the column names):

	year	award	amount
0	2015	NSF	7200.00
1	2016	DOE	9100.00
2	2016	CUNY	6300.00
3	2017	NSF	9400.00
4	2017	CUNY	4000.00
5	2018	CUNY	2700.00
6	2018	NSF	4300.00

a. Create a pandas DataFrame containing this data.

b. Write code which creates a histogram from the `amount` column. The bins for this histogram should be \$0-\$4000, \$4000- \$8000, and \$8000-\$12000 (and, of course, the height of each bar should be the number of rows where the amount is in that bar's range).

12. What will be displayed by the following function?

```
def recu(x):
   if x <= 2:
      return 8
   return x + recu(x-2)

def main():
   print(recu(5))

main()
```

13. What will be displayed by the following function?

```
def recu(x):
   if x <= 2:
      return 8

   return [x, recu(x-2)]

def main():
   print(recu(5))

main()
```

14. I have an attempt at a recursive version of the factorial function. It doesn't work. Explain what happens when I call, say, `fact3`. How can I fix it?

```
def fact(x):
   return x * fact(x-1)
```

15. I have the following function which computes the nth Fibonacci number. When I run `fib(100)`, the program appears to freeze up. Explain why.

```
def fib(n):
   if n <= 2:
      return 1
   else:
      return fib(n-1) + fib(n-2)
```

16. You can write a function to recursively **reverse** a list. The basic idea is that, for example, the reversal of `[6,1,4,7]` can be gotten by taking the first element (which would be `6`) and placing it after the reversal of `[1,4,7]` (which would be `[7,4,1]`).

    Complete the reversal function, based on this hint.

```
def r_reverse(x):
       if len(x) <= 1:
              ### FILL IN ###
       else:
              ### FILL IN ###
```

17. You can write a function to recursively count the number of times that `0` appears in a list. The basic idea is to look at the first element, and if it is a 0, add 1 to the number of zeros in the rest of the list (and otherwise just take the number of zeros in the rest of the list).

    Complete the following `count_zeros` function, based on this – you must write it recursively!

```
def count_zeros(x):
       if len(x) == 0:
              #####
       else:
              #####
              #####
```

```
#####
#####
```

18. Write a function called `my_sum`, which takes a list of numbers as input, and returns the sum of those numbers, using NO LOOPS: use *recursion* (and slicing) instead!

19. Write a *recursive* function called `dog`, which takes as input a positive float, and returns the number of times you can divide that number by 2 before you get an answer ≤ 1. For instance, `dog(9)` should return 4, because if you divide 9 by 2 you get 4.5; when you 4.5 by 2 you get 2.25; when you divide 2.25 by 2 you get 1.125; and finally, when you divide 1.125 by 2 you get 0.5625 which is ≤ 1. You should use NO LOOPS, and NO MATH functions other than `+,-,*,/` and comparisons (`<, >, <=, >=`).

20. Consider the following code, which performs a sort on a list called `numbers`:

```
for pos in range(len(numbers)):
    min_pos = pos

    for j in range(pos+1, len(numbers)):
        print("Hey")
        if numbers[j] < numbers[min_pos]:
            min_pos = j

    temp = numbers[pos]
    numbers[pos] = numbers[min_pos]
    numbers[min_pos] = temp
```

    The code also contains a print statement. If the length of `numbers` is 4, *exactly* how many times will `"Hey"` print? If the length of `numbers` is 400, *approximately* how many times will `"Hey"` print?

21. The function below is a broken version of the binary search.

```
def search(s_list, value):
    """Return True if value is in the sorted list called s_list, return False otherwise"""
    lower_bound = 0
    upper_bound = len(sorted_list) - 1

    while lower_bound <= upper_bound:
        halfway = (lower_bound + upper_bound)//2 #
        if s_list[halfway] == value:
            return True

   return False
```

    a. Give an example of a list `x` and a value `val` such that `search(x, val)` returns `True` as intended.

    b. Give an example of a list `x` and a value `val` such that `search(x, val)` does not work. Explain what happens instead.

22. Illustrate the steps of the *selection sort* working on the list [4, 12, 2, 8, 16, 1].

23. Suppose that I perform the *merge sort* on the list [4, 12, 2, 8, 16, 1, 9, 10]. The merge sort involves performing several different *merges* of sorted sub-lists. Write down all the different lists that are produced after a merge has taken place. (For example: [4, 12] is produced after merging the list [4] and the list [12], so that would be the first list. There should be 6 other lists of various sizes that are produced by merges; write them all down.)

24. If I perform mergesort on a list of 1000 numbers, approximately how many comparisons will be made between numbers as part of the merges? (Note: $1000 \approx 2^{10}$.)

25. Here's a slightly different version of selection sort:

```
def selectionSort(my_list):
   for fill_slot in range(len(my_list)-1,0,-1):
       pos_of_max=0
       for location in range(1,fill_slot+1):
           if ??????????????????????:
               pos_of_max = location

       temp = my_list[fill_slot]
       my_list[fill_slot] = my_list[pos_of_max]
```

```
    my_list[pos_of_max] = temp
```

Fill in the question mark line carefully.