CS 111 - Winter 2020
Quiz 5 Practice Questions

The quiz will be worth 30 points, and each point should correspond roughly to one minute of your time. This list of practice questions is not necessarily representative of the length of the quiz – it is hopefully longer to give you more practice, but there might be more questions in a given category on the actual quiz.

True/False questions (approximately 2 points)

1) Dictionary keys must be immutable.

2) Dictionary values must be immutable.

3) Using a for loop to loop over a dictionary iterates through the values.

4) Accessing the value for a given key in a dictionary is done using curly braces.

5) Looking up a key in a dictionary is fast.

6) Looking up a value in a dictionary is fast.

7) All keys in a dictionary must have the same type.

8) All values in a dictionary must have the same type.

9) The number of times n can be divided by 2 is $exp(n)$.

10) A recursive function must have exactly one base case.

Multiple-choice questions (approximately 3 points)

1) Which symbols are used to create a dictionary?
(a) `< >` **(b)** `< >` **(c)** `[ ]` **(d)** `{ }`

2) Approximately how many comparisons will linear search need to find an item in a list of 16 items, in the worst case?
(a) 4 **(b)** 16 **(c)** 64 **(d)** 256

3) Approximately how many comparisons will binary search need to find an item in a list of 16 items, in the worst case?
(a) 4 **(b)** 16 **(c)** 64 **(d)** 256

4) Approximately how many comparisons will linear search need to find an item in a list of 16 items, in the best case?
(a) 1 **(b)** 4 **(c)** 16 **(d)** 64

5) Approximately how many comparisons will binary search need to find an item in a list of 16 items, in the best case?
(a) 1 **(b)** 4 **(c)** 16 **(d)** 64

6) Approximately how many comparisons will selection sort need to sort a list of 16 items, in the worst case?
(a) 4 **(b)** 16 **(c)** 64 **(d)** 256

7) Approximately how many comparisons will merge sort need to sort a list of 16 items, in the worst case?
(a) 4 **(b)** 16 **(c)** 64 **(d)** 256

8) Approximately how many comparisons will selection sort need to sort a list of 16 items, in the best case?
(a) 4 **(b)** 16 **(c)** 64 **(d)** 256

9) Approximately how many comparisons will merge sort need to sort a list of 16 items, in the best case?
(a) 4 **(b)** 16 **(c)** 64 **(d)** 256

10) The recursive Fibonacci function is inefficient because
(a) it does many repeated computations
(b) recursion is inherently inefficient compared to iteration
(c) calculating Fibonacci numbers is "hard" (aka "intractable")
(d) it had a bug

11) How many steps would be needed to solve the Tower of Hanoi problem for a tower of size 5?
(a) 5 **(b)** 10 **(c)** 25 **(d)** 31

12) Which of the following is *not* true of the halting problem?
(a) It was studied by Alan Turing.
(b) It is harder than intractable.
(c) Someday a clever algorithm may be found to solve it.
(d) It involves a program that analyzes other programs.

Code prediction (approximately 4 points)

1) See the mini quiz from Lesson 24 (problem 3).

2) Consider the following recursive function.

```
1    def f(n):
2        print("n:", n)
3
4        if n <= 0:
5            return 0
6
```

```
 7        if n % 2 == 1:
 8            return f(n//2)
 9        else:
10            return n + f(n-1)
```

a) What is the output from calling `print(f(6))`?

b) What is the output from calling `print(f(5))`?

Short answer (approximately 3 points)

1) Place the algorithm classes in order from fastest to slowest: $n \lg n$, n, n^2, $\lg n$, 2^n.

2) List the two parts of a recursive function.

3) List two features of a recursive function that are necessary to avoid infinite recursion.

4) What is the worst-case running time of the sorting algorithm in the "Fixing bugs" problem? What is the best-case running time? Give your answer in terms of n, the number of items in the list.

Algorithm prediction (approximately 3 points)

1) See the mini quiz from Lesson 25 (problem 3).

2) See the mini quiz from Lesson 26 (problem 3).

3) See the mini quiz from Lesson 27 (problem 2).

4) See the mini quiz from Lesson 28, which didn't happen but is on Moodle (problem 2).

Rewriting code (approximately 3 points)

Rewrite the following iterative function to use recursion instead.

```
 1    def palindrome(s):
 2        """
 3        Returns True if the string s is a palindrome, and False otherwise.
 4
 5        Does not handle punctuation or spaces, so the string must
 6        have them match perfectly.
 7        """
 8        n = len(s)
 9        for i in range(n//2):
10            # If the character at position i and n-1-i match,
11            # it might be a palindrome.  If not, it's definitely not.
12            if s[i] != s[n-1-i]:
```

```
13                return False
14
15        # All characters matched!
16        return True
```

2) The following code prints information about each pet (stored as a name-age pair) in the dictionary.

```
nameToAgeMap = {"Sadie": 12,
                "Therese": 15,
                "Lulu": 7,
                "Hobbes": 7}

ageCounts = {}

for age in nameToAgeMap.values(): # iterates over sequence of values
    if age in ageCounts:
        ageCounts[age] += 1
    else:
        ageCounts[age] = 1

print("The age distributions are:")
for age in ageCounts:
    print("Age {0}: {1} pet(s)".format(age, ageCounts[age]))
```

Rewrite this code to iterate over the keys of `nameToAgeMap` instead of the values in the first loop, and to use `ageCounts.items()` in the second loop.

Fixing bugs (approximately 4 points)

The following function sorts a list using GnomeSort. Here is the definition of GnomeSort, taken from Wikipedia (which in turn cites Dickgrune.com):

"Here is how a garden gnome sorts a line of flower pots.

- Basically, he looks at the flower pot next to him and the previous one; if they are in the right order he steps one pot forward, otherwise, he swaps them and steps one pot backward.
- Boundary conditions: if there is no previous pot, he steps forwards; if there is no pot next to him, he is done."

Unfortunately, the following implementation has four bugs in it. Identify two of them and explain how to fix them.

Note that it might help to walk through an example, such as trying to sort the list `[1,4,2,9]`.

```
1    def gnomesort_buggy(mylist):
2        pos = 0
3        while pos <= len(mylist):
4            if mylist[pos] >= mylist[pos-1]:
5                pos += 1
6            else:
7                mylist[pos] = mylist[pos-1]
8                mylist[pos-1] = mylist[pos]
```

Writing code (approximately 8 points)

1) Write a function `characterFreq(s)` that takes in a string and returns a dictionary mapping characters to their frequencies within the string. You should convert each character to lowercase, so that `'A'` and `'a'` both count towards `'a'`.

2) Write a recursive function `max(mylist)` that finds the largest number in a list. (Hint: the max is the larger of the first item and the max of all of the other items.)

3) Write a recursive function `anagrams(s)` that returns a list of all possible anagrams of a given string `s`. For example, `anagrams("abc")` should return the list `['abc', 'bac', 'bca', 'acb', 'cab', 'cba']`.

4) Write a function `baseConversion(num, base)` that takes as input a number and a base, and then prints out the digits of the number in the new base. You should use recursion. Also, make sure to put a space between successive outputs, since bases greater than 10 will print out multi-character "digits". (Hint: Consider base 10. To get the rightmost digit of a base 10 number, look at the remainder after dividing by 10. For example, `153 % 10` is `3`. To get the remaining digits, you repeat the process on `15`, which is just 153 // 10. This process works for any base.)