

1. _____/15
2. _____/10
3. _____/10
4. _____/18
5. _____/8
6. _____/13
7. _____/15
8. _____/9
9. _____/1
10. _____/1

Total _____/100

This quiz is open book and open notes, but do not use a computer.

Please **write your name on the top of each page**. Answer all questions in the boxes provided.

1) Are each of the following True or False (15 points)

False

1.1. In Python the **values** of a dict must be immutable.

True

1.2. There exist problems that **cannot** be solved in Python **without** using either iteration or recursion.

False

1.3. Floating point arithmetic behaves exactly like normal arithmetic on real numbers.

False

1.4. On all inputs, a bisection search will run faster than a linear search.

False

1.5. Let L be a list, each element of which is a list of ints. In Python, the assignment statement $L[0][0] = 3$ mutates the list L .

2) What does the following code print? (10 points)

```
T = (0.1, 0.1)
x = 0.0
for i in range(len(T)):
    for j in T:
        x += i + j
        print x
print i
```

3) What does the following code print? (10 points)

```
def f(s):  
    if len(s) <= 1:  
        return s  
    return f(f(s[1:])) + s[0]  #Note double recursion  
  
print f('mat')  
print f('math')
```

4) Implement the body of the function specified in the box. (18 points)

```
def findAll(wordList, lStr):  
    """assumes: wordList is a list of words in lowercase.  
               lStr is a str of lowercase letters.  
               No letter occurs in lStr more than once  
    returns: a list of all the words in wordList that contain  
             each of the letters in lStr exactly once and no  
             letters not in lStr."""
```

5) The following code does not meet its specification. Correct it. (8 points)

```
def addVectors(v1, v2):
    """assumes v1 and v2 are lists of ints.
       Returns a list containing the pointwise sum of
       the elements in v1 and v2. For example,
       addVectors([4,5], [1,2,3]) returns [5,7,3], and
       addVectors([], []) returns []. Does not modify inputs."""
    if len(v1) > len(v2):
        result = v1
        other = v2
    else:
        result = v2
        other = v1
    for i in range(len(other)):
        result[i] += other[i]
    return result
```

6) Consider the following code:

```
def f(s, d):
    for k in d.keys():
        d[k] = 0
    for c in s:
        if c in d:
            d[c] += 1
        else: d[c] = 0
    return d

def addUp(d):
    result = 0
    for k in d:
        result += d[k]
    return result

d1 = {}
d2 = d1
d1 = f('abbc', d1)
print addUp(d1)
d2 = f('bbcaa', d2)
print addUp(d2)
print f('', {})
print result
```

6.1) What does it print? (9 points)

6.2) Does it terminate normally? Why or why not? (4 points)

7) Consider the following code:

```
def logBase2(n):  
    """assumes that n is a positive int  
       returns a float that approximates the log base 2 of n"""  
    import math  
    return math.log(n, 2)  
  
def f(n):  
    """assumes n is an int"""  
    if n < 1:  
        return  
    curDigit = int(logBase2(n))  
    ans = 'n = '  
    while curDigit >= 0:  
        if n%(2**curDigit) < n:  
            ans = ans + '1'  
            n = n - 2**curDigit  
        else:  
            ans = ans + '0'  
        curDigit -= 1  
    return ans  
  
for i in range(3):  
    print f(i)
```

7.1) What does it print? (10 points)

7.2) Under the assumption that logBase2 is $O(n)$, what is the order (use big Oh notation) of f ? (5 points)

8) Next to each item in the left column write the letter labeling the item in the right column that best matches the item in the left column. No item in the right column should be used more than once. (9 points)

e, f

Big O notation

a) induction

d, b, c

Newton's method

b) upper bound

a

recursion

c) lower bound

d) approximation

e) expected running time

f) exponential

9. Do you think that the lectures are too slow paced, too fast paced, about right? (1 point)

10. Do you think that the problem sets are too easy, too hard, about right? (1 point)

MIT OpenCourseWare
<http://ocw.mit.edu>

6.00SC Introduction to Computer Science and Programming
Spring 2011

For information about citing these materials or our Terms of Use, visit: <http://ocw.mit.edu/terms>.