

-----  
Name

1. \_\_\_\_\_/18

2. \_\_\_\_\_/3

3. \_\_\_\_\_/20

4. \_\_\_\_\_/24

5. \_\_\_\_\_/10

6. \_\_\_\_\_/10

7. \_\_\_\_\_/15

-----  
Athena User Name-----  
Recitation hour

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**, and your user name and the hour of the recitation you attend on the first page. Answer all questions in the boxes provided.

1) Are each of the following True or False (18 points)

 T

1.1. **Any** program that can be written using only the basic arithmetic operators, assignment, and conditionals (and no function calls or iteration) will run in constant time.

 F

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

 F

1.3. In Python, dictionaries are **immutable**.

 F

1.4. Recursive solutions to problems are usually **more** computationally efficient than iterative ones.

 F

1.5. When Newton-Raphson is used to find the root of a polynomial, the **order** of complexity is linear in the number of terms in the polynomial.

 F

1.6. Given the same input, a program that is  $O(n^2)$  will **always** take longer to run than a program that is  $O(\log n)$ .

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

```
y = 10.0
x = 1.0
for i in range(10):
    y = y - 1.0
    x = x - 0.1
print x == y
```

False

3) Write a Python function that meets the following specification. (20 points)

```
def get_word_score(word, charVals):
    """If word is not of type str, returns -1, otherwise
    Returns the score for word, where
        if c, a char, is a key in charVals,
            the value of c is the value associated with that key
            otherwise the value associated with c is zero
        the score of a word is the product of the values of the chars
        in word"""

    if type(word) != str:
        return -1
    if len(word) == 0:          # No points taken off if this edge case
        return 0                # is missing
    score = 1
    for c in word:
        if c in charVals:      # Can also use
            score *= charVals[c] # score *= charVals.get(c, 0)
        else:                  # instead of this entire embedded block
            return 0
    return score
```

4) Consider representing polynomials using a **dictionary**. The keys are non-negative ints representing the exponents, and the values are floats representing the coefficients for the term with that exponent. Complete the implementation of the function below. (24 points)

```
def compute_deriv(poly, x):
    """Computes and returns the derivative (a float) of the
       polynomial at point x."""

    deriv = {}
    for exponent in poly:
        # exponent != 0 check only necessary because x = 0 would cause a
        # 0 raised to a negative power error
        # No points taken off if this check is missing
        if exponent != 0:
            deriv[exponent - 1] = exponent*poly[exponent]
    result = 0
    for deriv_exponent in deriv:
        result += deriv[deriv_exponent]*(x**deriv_exponent)
    return result
```

5) Consider the function definition of `squareRootBi` shown here. This code fails to meet its specification. Point out the mistake and correct it. (10 points)

```
def squareRootBi(x, epsilon):
    """Assumes x and epsilon are floats.
       Assumes x >= 0 and epsilon > 0
       Return y s.t. y*y is within epsilon of x"""
    assert x >= 0, 'x must be non-negative'
    assert epsilon > 0, 'epsilon must be positive'
    low = -x
    high = 0
    guess = low
    ctr = 1
    while abs(guess**2 - x) > epsilon and ctr <= 100:
        if guess**2 > x:
            low = guess
        else:
            high = guess
        guess = (low + high)/2.0
        ctr += 1
    assert ctr <= 100, 'Iteration count exceeded'
    return guess
```

The mistake is that if  $x < 1$ , the negative square root of  $x$  is less than  $-x$ . So when we first set `low` and `high` to  $-x$  and  $0$ , they do not contain the square root.

To correct this, we can simply change

```
low = -x
```

to

```
low = min(-x, -1)
```

6 What does the following print? (10 points)

```
t = (1,2,3)
q = t
d = {1:t, 2:q}
e = d
t = t + (4,)
print t == q
print d == e
d[1] = q
print d == e
```

```
False
True
True
```

7) Consider the following code:

```
def f(L):  
    result = []  
    for e in L:  
        if type(e) == list:  
            return f(e)  
        else:  
            result.append(e)  
    return result  
  
print f([1, [2, ['a', 'b']], (3, 4)])
```

7.1. What does it print? (10 points)

```
['a', 'b']
```

7.2. Which of the following statements best describes the behavior of f?

- a) f is constant in the number of elements in L, i.e., in len(L).
- b) f is linear in the number of elements in L, i.e., in len(L).
- c) f is quadratic in the number of elements in L, i.e., in len(L).
- d) None of the above

(5 points)

d

8. Do you think that the lectures are too slow paced, too fast paced, about right? (0 points)

9. Do you think that the problem sets are too easy, too hard, about right? (0 points)