This quiz is open book and open notes, but do not use a computer. **You have 120 minutes.**

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 (24 points)

F 1.1. In a Python program, you can have a variable named return, e.g., return = 5.

T 1.2. The following expression has correct syntax: x, y = 1, 2.

F 1.3. A program that keeps running and does not stop is an example of a syntax error.

F 1.4. A function is an example of decomposition but not abstraction.

F 1.5 The dictionary {'a': '1', 'b': '2', 'c': '3'} has a mapping of string:int

T 1.6 assert statements in your code are a way of checking that preconditions to functions are met.

T 1.7 Given a list l = ['f', 'b'] the statement l[1] = 'c' will mutate list l.

F 1.8 Given a list l = ['f', 'b'] the statement l[1][0] = 'c' will mutate the second string object of l.
2) What does the following code print? (4 points)

```python
a = '3'
b = '-3'
print a+b
```

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

```python
def testprog(x, y):
    temp = x
    x = y
    y = temp
    print x

x = 3
y = -3
print x
print x
```

3
-3
3
4) What is the value of variable z in each of the following code snippets: (12 points)

```python
x = 3
y = 4
z = x/y
```

0

```python
x = (1, 'rosemary')
y = (2, 'thyme')
if( x[0]*y[0] > 0 ):
    z = "Fight crime"
else:
    z = "Are delicious"
```

"Fight crime"

```python
x = [1,2,3]
x.append(0)
z = x[1:3]
z.remove(2)
```

[3]

5) Which part of this code should be rewritten as a function? Write the function on the next page. Your function should have as much functionality as possible. *Hint: Your*
function can return a value if you choose. (10 points)

# This program is a simple calculator. Program loops until
# user stops play. User enters three inputs: one number, an
# operation, another number. Prints the result of the
# operation on the numbers and the number of computations.

again = 'y'
umtimes = 0

while again == 'y':
    value1 = float(raw_input('Enter the first number: '))
    operation = raw_input('Operation? Enter +, -, * or /: ')
    value2 = float(raw_input('Enter the second number: '))

    if operation == '+':
        result = value1 + value2
        print '=' + str(result)
        numtimes += 1
        print 'You played ' + str(numtimes) + ' times'

    if operation == '-':
        result = value1 - value2
        print '=' + str(result)
        numtimes += 1
        print 'You played ' + str(numtimes) + ' times'

    if operation == '*':
        result = value1 * value2
        print '=' + str(result)
        numtimes += 1
        print 'You played ' + str(numtimes) + ' times'

    if operation == '/':
        result = value1 / value2
        print '=' + str(result)
        numtimes += 1
        print 'You played ' + str(numtimes) + ' times'

    again = raw_input('Play (y or n)? ')
THE FUNCTION:

```python
def perform_op(numtimes):
    # Get the two numbers and the operation to perform
    value1 = float(raw_input('Enter the first number: '))
    operation = raw_input('Operation? Enter +, -, * or /: ')
    value2 = float(raw_input('Enter the second number: '))

    # Print off the result of the operation and
    # How many times the player has played.
    results = {'+': value1+value2,
               '-': value1-value2,
               '*': value1*value2,
               '/': value1/value2}
    print ' = ' + str(results[op])
    numtimes += 1
    print 'You played ' + str(numtimes) + ' time'
    return numtimes
```

THE WHILE LOOP BECOMES:

```python
while again == 'y':
    numtimes = perform_op(numtimes)
    again = raw_input('Play (y or n)? ')
```
6) Write a program that asks the user for a positive integer K and checks if K is a triangular number and prints YES or NO. A triangular number is a number obtained by the continued summation of natural numbers. For example, 1, 1+2, 1+2+3, 1+2+3+4, etc., corresponding to 1, 3, 6, 10, etc., are triangular numbers. What is the asymptotic complexity of your implementation in terms of K? Your code will be graded for correctness as well as efficiency. (20 points)

```python
def is_triangular(K):
    sum = 0
    for i in xrange(1, K+1):
        sum += i
        if K == sum:
            return True
        elif K < sum:
            return False
```

The asymptotic complexity of this algorithm is $O(K^{0.5})$ because the loop will end with a return of True/False after roughly $n = (2K)^{0.5}$ iterations – more precisely $K = n(n+1)/2$.

A different more efficient strategy asymptotically speaking is to directly use the fact that K needs to be $n(n+1)/2$ where n is an integer for K to be a perfect number. Given K, solve for n – we have a quadratic equation. This requires the computation of a square root for K, which can be done using binary search and $O(\log K)$ iterations.
7) Write a recursive function that calculates the greatest common divisor of two positive integers. Your function will use Euclid’s algorithm. For example, the greatest common divisor (gcd) between \(a=20\) and \(b=12\) is calculated as:

\[
gcd(20, 12) \text{ is the same as } gcd(12, 20 \mod 12) = gcd(12, 8) \]
\[
gcd(12, 8) \text{ is the same as } gcd(8, 12 \mod 8) = gcd(8, 4) \]
\[
gcd(8, 4) \text{ is the same as } gcd(4, 8 \mod 4) = gcd(4, 0) \]

The gcd is found (and the gcd is equal to \(a\)) when we reach \(0\) for \(b\). **Hint: remember the mod symbol is \% in Python.** (20 points)

```python
def gcd(a, b):
    if b == 0:
        return a
    return gcd(b, a % b)
```