Name:

E-mail ID:

On my honor, I pledge that I have neither given nor received help on this test.

Signature:

Test rules

- Print your name, id, and pledge as requested.
- This pledged exam is closed textbook. The only device you may access during the test is your own laptop.
- You are not allowed to access class examples or your own past assignments during the test; i.e., the only Python code you may access or view are ones that you develop for this test.
- The only windows that can be open on your computer are PyCharm and a single browser with tabs only open to the class website.
- PyCharm can be used only for developing the Python files to be submitted. It cannot be used for the true-false and short answer questions.
- Code should compile and demonstrate proper programming style; e.g., header comments, whitespace, identifier naming, etc.

Page 2:	**/ 9**
Page 3:	**/ 6**
Page 4:	**/ 10**
Total :	**/ 25**

1. (9 points) Consider the following code segment. In answering the below true and false questions.

```
 1: def f( x ) :
 2:     y = x + x
 3:     a = y
 4:     print( y )
 5:
 6: def g() :
 7:     return 1
 8:     return 2
 9:
10: a = 2
11: b = f( a )
12: print( a )
13: c = g()
14: print( f.y )
```

a. True or false: `y` is a variable local to function `f`.

b. True or false: `y` is the return value for function `f`.

c. True or false: in line 1, `x` is used as an argument.

d. True or false: in line 2, `x + x` is an invocation.

e. True or false: lines 3 and 10 reference the same variable `a`.

f. True or false: in line 11, `a` is used an argument

g. True or false: in line 12, `print( a )` is an invocation.

h. True or false: in line 13, c is initialized to 1.

i. True or false: in line 14, 4 is printed.

2. (6 points) Suppose the following five function definitions are in effect

```
def e( x ) :
   x = 3

def f( x ) :
   x.append( 3 )

def g( x ) :
   y = x
   y.append( 3 )

def h( x ) :
   return e( x )

def j( x ) :
   y = [ x ]
   f( y )
   return y
```

a. What is the output of the following code segment?

```
a = 1
e( a )
print( a )
```

b. What is the output of the following code segment?

```
x = 1
e( x )
print( x )
```

c. What is the output of the following code segment?

```
a = []
f( a )
print( a )
```

d. What is the output of the following code segment?

```
a = []
g( a )
print( a )
```

e. What is the output of the following code segment?

```
a = 1
b = h( 3 )
print( b )
```

f. What is the output of the following code segment?

```
a = [ 1, 2 ]
b = j( a )
print( b )
```

3. (2 points) What should the comment be for describing function `f()` with its numeric parameters `x`, `y`, and `z`?

```
def f( x, y, z ) :
   b1 = x != y
   b2 = x != z
   b3 = y != z
   b = b1 and b2 and b3
   return b
```

4. (2 points) What should the comment be for describing function `f()` with its nonempty numeric list parameter `x`.

```
def f( x ) :
   for v in x :
      if ( v % 2 == 0 ) :
         return True
   return False
```

5. (2 points) What should the comment be for describing function `f()` with its nonempty numeric list parameter `x`?

```
def f( x ) :
   for v in x :
      if ( v % 2 != 0 ) :
         return False
   return True
```

6. (2 points) What should the comment be for describing function `f()` with its nonempty numeric list parameter `x`?

```
def f( x ) :
   m1 = min( x )
   m2 = max( x )
   b = ( m1 == m2 )
   return b
```

7. (2 points) What should the comment be for describing function `f()` with its nonempty numeric list parameter `x` and its numeric parameter `k`?

```
def f( x, k ) :   :
   a = 0
   for v in x :
      if ( v < k ) :
         a = a + 1
      elif ( v > k ) :
         a = a + 1
   n = len( x )
   return (n - a)
```

Part II Module implementation

8. (15 points) Develop module `a.py`. The module defines a single function `f()`. The function has no parameters and does not have a `return` statement. The function prints *success* and nothing else. Also, develop a program `atest.py`. The only action of the tester is to invoke function `f()` exactly once. The tester should produce output

```
success
```

9. (15 points) Develop module `b.py`. The module defines a single function `f()`. The function has four numeric parameters `a`, `b`, `c`, and `d`. The function does not print any output. If the sum of the parameters is negative, the function returns `-1`; if the sum of the parameters is zero, the function returns `0`; and if the sum of the parameters is positive, the function returns `1`. A tester `btest.py` for function `f()` is available. A run of the tester should produce output

```
f(  1, -4,  4, -1 ): 0
f(  0,  2, -4,  1 ): -1
f( -1, -1,  5,  0 ): 1
```

10. (15 points) Develop module `c.py`. The module defines a single function `f()`. The function has one string parameter `w`. The function does not print any output. The function returns whether `w` contains a lowercase vowel. A tester `ctest.py` for function `f()` is available. A run of the tester should produce output

```
f( 'AEIOU' ): False
f( 'styx' ): False
f( '' ): False
f( 'bad' ): True
f( 'bed' ): True
f( 'bid' ): True
f( 'bod' ): True
f( 'bud' ): True
```

11. (15 points) Develop module `d.py`. The module defines a single function `f()`. The function has one numeric list parameter `x`. The function does not print any output. The function returns a new list whose values are the non-negative values of `x` in sorted order. A tester `dtest.py` for function `f()` is available. A run of of the tester should produce output

```
f( [ ] ): []
f( [ -3, -1, -4 ] ): []
f( [ 0,  0 ] ): [0, 0]
f( [ -3,  1, -4,  5 ] ] ): [1, 5]
f( [ 3,  1,  4,  5,  9 ] ): [1, 3, 4, 5, 9]
```

12. (15 points) Develop module `e.py`. The module defines a single function `f()`. The function has one numeric list parameter `x`. The function does not print any output. The function returns a new dictionary. The keys for the dictionary are the element values of `x`. A negative `x` element value is mapped to `'-'`; a positive `x` element value is mapped to `'+'`; and a zero `x` element value is mapped to `'0'`. A run of of the tester should produce output

```
f( [-3, -1, -4] ):  {-4: '-', -3: '-', -1: '-'}
f( [0,  0] ):  {0: '0'}
f( [1, -1,  0] ):  {0: '0', 1: '+', -1: '-'}
f( [-3,  1, -4,  5] ):  {1: '+', -4: '-', -3: '-', 5: '+'}
f( [3,  0,  4,  5,  -9 ] ):  {0: '0', 3: '+', 4: '+', 5: '+', -9: '-'}
```