**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:** | **/ 10** |
| **Page 3:** | **/ 8** |
| **Page 4:** | **/ 12** |
| **Total :** | **/ 30** |

1. (10 points) Consider the following code segment. In answering the below true and false questions.

```
def f( x ) :
    y = 10 * x
    print( y )

a = 2
b = f( a )
print( 'y =', y )
```

a. True or false: `x` is called the parameter of `f`.

b. True or false: `x` is called the argument of `f`.

c. True or false: `x` is called the input of `f`.

d. True or false: `f` does not have a return value.

e. True or false: the return value of `f` is `None`.

f. True or false: `f( a )` is an invocation.

g. True or false: `a` is a local variable of `f`.

h. True or false: `x` is a local variable of `f`.

i. True or false: `y` is a local variable of `f`.

j. True or false: the statement `print( 'y =', y )` causes `y = 20` to be printed.

2. (8 points) Suppose the following four function definitions are in effect

```
def s( a ) :
    a = 1112

def t( a ) :
    a = 1112
    return a
```

```
def u( a ) :
    a[ 0 ] = 1112

def v( a ) :
    a = [ 1112 ]
    return a
```

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

```
x = 1
s( x )
print( x )
```

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

```
a = 1
s( a )
print( a )
```

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

```
x = 1
t( x )
print( x )
```

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

```
a = 1
t( a )
print( a )
```

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

```
x = 1
x = t( x )
print( x )
```

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

```
x = [ 3, 1, 4, 1 ]
u( x )
print( x[ 0 ] )
```

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

```
x = [ 3, 1, 4, 1 ]
v( x )
print( x[ 0 ] )
```

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

```
x = [ 3, 1, 4, 1 ]
x = v( x )
print( x[ 0 ] )
```

3. (3 points) What should the comment be for describing function `f()`?

```
def f( x, y, z ) :
    t1 = type( x )
    t2 = type( y )
    t3 = type( z )
    return ( (t1 == t2 ) and ( t2 == t3 ) )
```

4. (3 points) What should the comment be for describing function `f()`?

```
def f( x ) :
    n = len( x )
    for i in range( 0, n ) :
        if ( x[ i ] < 0 ) :
            x[ i ] = -x[ i ]
```

5. (3 points) What should the comment be for describing function `f()`?

```
def f( x ) :
    b1, b2, b3 = False, False, False
    n = len( x )
    for i in range( 0, n ) :
        if ( x[ i ] < 0 ) :
            b1 = True
        elif ( x[ i ] == 0 ) :
            b2 = True
        else :
            b3 = True
    b = b1 and b2 and b3
    return b
```

6. (3 points) What should the comment be for describing function `f()`?

```
def f( x ) :
    b1, b2, b3 = False, False, False
    n = len( x )
    for i in range( 0, n ) :
        b1 = b1 or ( x[ i ] <  0 )
        b2 = b2 or ( x[ i ] == 0 )
        b3 = b3 or ( x[ i ] >  0 )
    b = b1 and b2 and b3
    return b
```

## Part II Module implementation

7. (10 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 *your* email id and nothing else. Also develop a program `atest.py`. The only action of the tester is to invoke function `f()` exactly once. Suppose the email id of the code developer for `a.py` is `mst3k`. The output of the tester should be

```
mst3k
```

8. (10 points) Develop module `b.py`. The module defines a single function `f()`. The function has four parameters `a`, `b`, `c`, and `v` that are to be numeric values. The function does not print any output. The function returns the value of $a \times v^2 + b \times v + c$. A tester `btest.py` for function `f()` is available. A run of the tester should produce output

```
32.25
45.125
37.516000000000005
```

9. (10 points) Develop module `c.py`. The module defines a single function `f()`. The function has one parameter `x` that is to be a list of strings. The function does not print any output. The function returns the length of the longest string in `x`. A tester `ctest.py` for function `f()` is available. A run of the tester should produce output

```
5
6
7
```

10. (10 points) Develop module `d.py`. The module defines a single function `f()`. The function has two parameters `b` and `c` that are to be numeric lists. The function does not print any output. The function returns the *inner product* of `b` and `c`, where

    - If `b` and `c` have different lengths, the inner product is `None`.
    - If `b` and `c` have the same length, then the inner product is

    $$(b[\,0\,] \times c[\,0\,]) + (b[\,1\,] \times c[\,1\,]) + \ldots + (b[\,n-1\,] \times c[\,n-1\,])$$

    where `n` is the length of the lists. A tester `dtest.py` for function `f()` is available. A run of of the tester should produce output

```
38
None
55
```

11. (10 points) Develop module `e.py`. The module defines a single function `f()`. The function has one parameter `s` whose value is a string containing zero or more numerical values. The function does not print any output. The function returns the numeric list corresponding to `s`. For example, if `x = '1.2 3.4 5.6'` then `f(x)` returns `[1.2, 3.4, 5.6 ]`. A tester `etest.py` for function `f()` is available. A run of of the tester should produce output

```
[3.0, 1.0, 4.0, 1.0]
[5.0, 9.0, 2.0]
[1.25, 2.5, 3.75, 4.0, 5.25]
```

12. (10 points) Develop module `f.py`. The module defines a single function `f()`. The function has two parameters `x` and `y` whose values are lists. The function does not print any output. The function returns as a list one copy of each value in `x` that is not in `y`. For example, the following code segment

```
x1 = [ 3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5, 8, 9 ]
y1 = [ 2, 7, 1, 8, 2, 8, 1, 8, 2, 8, 4, 5 ]
u1 = f.f( x1, y1 )
```

sets `u1` to `[3, 9, 6]`. A tester `ftest.py` for function `f()` is available. A run of of the tester should produce output

```
[3, 9, 6]
['s', 'i', 'g']
[]
```

13. (10 points) Develop module `g.py`. The module defines a single function `f()`. The function has one parameter `m` whose value is to be a `dict`. The function does not print any output. The function returns whether each key in `m` maps to a different value. For example, the following code segment

```
abc = { 'A' : 'apple', 'B' : 'banana', 'C' : 'cherry' }
roman = { 'i' : 1, 'I' : 1, 'v' : 5, 'V' : 5, 'x' : 10, 'X' : 10 }
b1 = g.f( abc )
b2 = g.f( roman )
```

sets `b1` to `True` because each key in `abc` maps to a unique value and set `b2` to `False` as different keys in `roman` are mapped to the same value. A tester `gtest.py` for function `f()` is available. A run of of the tester should produce output

```
True
True
False
```