Short Answer Questions (40 points)

1. TRUE FALSE You have very legibly printed your name and email id below.

   Name = _____ EMAILD = _____

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

3. TRUE FALSE All Python modules are programs.

4. TRUE FALSE Python function parameters are named in the function definition.

5. TRUE FALSE Python function parameters are named in a function invocation.

6. TRUE FALSE A Python function parameter can also act as a function argument.

7. TRUE FALSE Python function arguments are given in a function invocation.

8. TRUE FALSE If a Python function invocation does not supply enough values for the function, Python supplies None for each of the missing values.

9. TRUE FALSE No matter what Python function `f()` does, the output of the following Python code segment is always 123.

```
x = 123
f( x )
print( x )
```

10. TRUE FALSE No matter what Python function `f()` does, the output of the following Python code segment is always 123.

```
x = 123
x = f( x )
print( x )
```

11. TRUE FALSE Assume that Python function `f()` does not have a return statement, then no matter what `f()` does, the output of the following code segment is always None.

```
x = 123
x = f( x )
print( x )
```

12. TRUE FALSE No matter what Python function `f()` does, the output of the following Python code segment is always `[ 1 ]`.

```
x = [ 1 ]
f( x )
print( x )
```

13. TRUE FALSE No matter what Python function `f()` does, the output of the following Python code segment is always `[ 1 ]`.

```
x = [ 1 ]
x = f( x )
print( x )
```

14. TRUE FALSE All Python function invocations have a return value.

15. TRUE FALSE The only purpose of a Python function return statement is to indicate that the function execution has been completed.

16. TRUE FALSE Although local variables only exist during the execution of their function, their values survive from invocation to invocation.

17. TRUE FALSE The following function definition always correctly determines whether x is equal to the minimum of x, y, and z.

```
def f( x, y, z ) :
    if ( (x <= y ) and ( y <= z ) ) :
        return True
    else :
        return False
```

18. TRUE FALSE A *symmetric* dictionary d is one that follows this rule: if d[x] equals y, then it also the case that d[y] equals x. The following function always correctly tests whether d is symmetric.

```
def f( d ) :
    for x in d.keys() :
        y = d[ x ]
        if ( d[ y ] == x ) :
            return True
        else :
            return False
```

19. TRUE FALSE The following function always correctly tests whether list x is in sorted order.

```
def f( x ) :
    prev = x[ 0 ]
    for v in x :
        if ( prev > v ) :
            return False
        else :
            prev = v
    return True
```

20. TRUE FALSE Consider function `f()`.

```
def f( x, y ) :
    remember = x
    x = y
    y = remember
    return x, y
```

The below statement always correctly swaps the values of `a` and `b`.

```
a, b = f( a, b )
```

21. TRUE FALSE All Python function invocations require the use of parentheses.

Part 2: Problem solving

22. (20 points) Module *ncwl.py* defines functions `characters()`, `words()`, `lines()`, and `cwl()`. Each of the functions has a single parameter `s`. Functions `characters()`, `words()`, and `lines()` return an integer; function `cwl()` returns a three-element list of integers.

    - Function `characters()` returns the number of characters in `s`.
    - Function `words()` returns the number of words in `s`.
    - Function `lines()` returns the number of lines in `s`.
    - Function `cwl()` returns a three-element list of integers; that are respectively the number of characters, words, and lines in `s`.

    A simple tester *ntest.py* is available. The output of the tester should be

```
29 56 415 0
6 11 69 0
1 3 14 1
[29, 6, 1] [56, 11, 3] [415, 69, 14] [0, 0, 1]
```

23. (20 points) Module *unique.py* defines two functions `row()` and `table()`. Neither function should modify its list parameter.

    - Function `row()` takes a single list parameter `r`. The function returns the elements of `r` that are not duplicated elsewhere in `r`; i.e., they occur exactly once in all of `r`. For example, the following function invocation

      ```
      unique.row( [ 2, 4, 6, 8, 2, 8, 6, 3 ] )
      ```

      returns `[4, 3]`.
    - Function `table()` takes a single dataset parameter `d`; that is `d` is a list of rows. The function returns the cells of `d` that are not duplicated elsewhere in `d`; i.e., they occur exactly once in all of `d`. For example, the following function invocation

      ```
      unique.table( [ [ 2, 4, 6, 8, 2, 8, 6, 3 ], [ 3, 1, 5 ] ] )
      ```

      returns `[4, 1, 5]`.

    A simple tester *utest.py* is available. The output of the tester should be

```
['A', 'C', 'D', 'F', 'G', 'H', 'K']
['A', 'B', 'O']
['A', 'G', 'K', 'N']
['A', 'B', 'E', 'N']

['I', 'J', 'O']
['D', 'E']
['B', 'J']
['C', 'H', 'K', 'L', 'N', 'P']
```

An examination of the tester indicates it prints the solutions to `row()` and `table()` in sorted order. It does so, because different approaches to the problem can produce different orderings for the solutions.

24. (20 points) Module *primal.py* defines two functions `is_factor()` and `is_prime()`.

- Function `is_factor()` takes two integer parameters `x` and `y`. The function returns `True` or `False` whether `y` is a factor of `x`; that is whether the remainder of `x` divided by `y` is 0.
- Function `is_prime()` takes a single integer parameter `x`. The function returns `True` or `False` whether `x` is a prime number. There are lots of definitions for a number being prime. The one that might inspire a solution for the function is that `x` is prime if none of the values in the interval `2 … x-1` are factors of `x`.

A simple tester *ptest.py* is available. The output of the tester should be

```
is_factor( 6 , 2 ): True
is_factor( 10 , 10 ): True
is_factor( 7 , 3 ): False
is_factor( 9 , 8 ): False

is_prime( 2 ): True
is_prime( 103 ): True
is_prime( 10 ): False
is_prime( 45 ): False
```

Notices

- Based on your past educational achievements, I expect you to do well on this test.
- You can answer the questions in any order that you want.

Test rules

- This pledged exam is closed notes. The only device you may access during the test is your laptop.
- Do not access class examples or your own past assignments during the test; that is, the only code you may access or view are ones that you develop for this test.
- The only windows to be open on your computer are PyCharm and a single browser with tabs reachable from the class website.
- PyCharm can be used for developing the modules to be submitted. It **cannot be used** for the short answer questions.
- Functions should demonstrate follow style rules; e.g., header comments, whitespace, identifier naming, etc.
- Whether a function is testable is important.
- Only do what is requested. None of the functions you develop should get input or produce output. Unless indicated, functions should not modify their parameters in any way.
- Any form of cheating on a test can result in expulsion from the class and the incident being referred to the Honor Committee.