SOLUTION KEY

1. (7.0 points) **WWPD: Control**

What Would Python Display?
Assume the following code has been executed.

```
def enigma(a, b, c, d):
    if a < 0:
        return None
    while b < c:
        c = c - 1
        d = d + 3
    return c > d

print(enigma(1, 2, 3, 4)) # (a)
print(enigma(1, 2, 3, -4))  # (b) & (c)
print(enigma(-4, 8, 4, 1)) # (d)
print(enigma(1, 2, 3, -1))   # (e)
```

(a) (1.5 pt) What value is printed?

- `True`
- ~~`False`~~
- `None`

(b) (1.5 pt) What value is printed?

- ~~`True`~~
- `False`
- `None`

(c) (1.5 pt) What value is bound to `a` at the *end* of the second function call to enigma at (c)?

1

(d) (1.5 pt) What value is printed?

- `True`
- `False`
- ~~`None`~~

(e) (1.0 pt) What value is printed?

SOLUTION KEY

- True
- ~~False~~
- None

SOLUTION KEY

2. (10.0 points) **Finding Prime Numbers using Hailstone's 3x+1**

```
from math import sqrt

def is_prime(n):
    """Return True if N is prime."""
    return n > 1 and smallest_factor(n) == n

def smallest_factor(n):
    """Returns the smallest value k>1 that evenly divides N."""
    f = n
    x = 2
    while x <= round(sqrt(n)+1):
        if (n % x) == 0 (a):
            f = x
            break
        x = x + 1
    return f

def three_x_plus_1(x):
    """Compute 3x+1 for any number, given N >= 1.
    >>> three_x_plus_1(2)
    7
    >>> three_x_plus_1 (6)
    19
    """
    return 3*x+1 (b)

def print_prime(n, f):
    """Print out all prime numbers generated by calling function f on the range
of 1 to n.
    >>> print_prime(2, three_x_plus_1)
    7
    >>> print_prime(3, three_x_plus_1)
    7
    >>> print_prime(4, three_x_plus_1)
    7
    13
    >>> print_prime(5, three_x_plus_1)
    7
    13
```

SOLUTION KEY

```
>>> print_prime(13, three_x_plus_1)
7
13
19
22
31
37
"""
x = 1  (c)
while x <= n:
    y = f(x)
    if is_prime(y)  (d):
        print(y)
    x += 1  (e)
```

(a) (2.0 pt) What should fill in blank (a)?

```
0
```

(b) (2.0 pt) What should fill in blank (b)?

```
3*x+1
```

(c) (2.0 pt) What should fill in blank (c)?

```
1
```

(d) (2.0 pt) What should fill in blank (d)?

```
is_prime(y)
```

(e) (2.0 pt) What should fill in blank (e)?

SOLUTION KEY

```
x += 1
```

3. (10.0 points) **Function Equivalence: Multiplication**

Definition. Two functions f and g have identical behavior if f(x) and g(x) return equal values or return functions with identical behavior, for every x that does not cause an error.

```
from operator import mul

def double(x):
    return x * 2

def triple(x):
    return x * 3

def enigma (y):
    return double(triple(y)) + triple(triple(y))

def multiply_by(a):
    def slow_multiplication(b):
        sum = 0
        x = 0
        while (x < a):
            sum = b + sum
            x = x + 1
        return sum
    return slow_multiplication
```

NOTE: the function `mul(x, y)` computes `x*y`

(a) (2.0 pt) The result of evaluating `` `multiply_by(2)(5)` `` has identical behavior to the result of evaluating the expression … (**check all that apply**)

- `enigma(6)`
- `enigma(2)`
- `enigma(5)`
- ~~`double(5)`~~
- `double(2)`
- `triple`
- `triple(5)`
- ~~`mul(5, 2)`~~
- ~~`mul(2, 5)`~~

SOLUTION KEY

- `mul(3, 4)`
- `mul`
- `make_power_of(1)(11)`

(b) (2.0 pt) The result of evaluating `` `multiply_by(3)(5)` `` has identical behavior to the result of evaluating the expression … (**check all that apply**)

- ~~`enigma(1)`~~
- `enigma(2)`
- `enigma(3)`
- `engima`
- ~~`triple(5)`~~
- `triple(3)`
- `triple`
- ~~`mul(3, 5)`~~
- ~~`mul(15, 1)`~~
- `mul(2, 10)`
- `mul`

(c) (2.0 pt) The result of evaluating `` `multiply_by(2)` `` has identical behavior to the result of evaluating the expression … (**check all that apply**)

- `enigma(10)`
- `enigma`
- `double(2)`
- `double(5)`
- ~~`double`~~
- `mul(2, 5)`
- `mul(1, 10)`
- `mul`

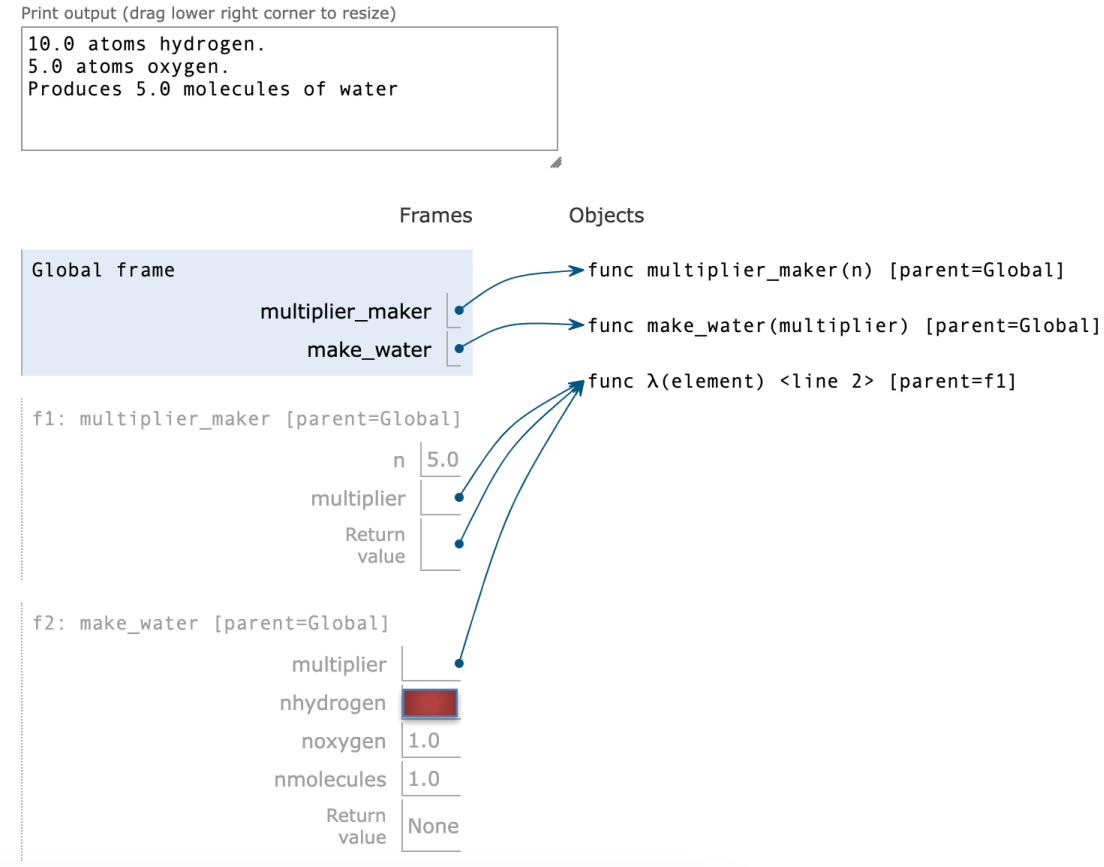
(d) (2.0 pt) What is the type of the return value for the function `` `multiply_by' ``?

- ~~`Function`~~
- `Integer`
- `Float`
- `String`
- `None`

SOLUTION KEY

4. (8.0 points) **Chem 105**

Consider the environment diagram (and print output) below, followed by the code that generated it. The red box has been placed to hide answers




```
def multiplier_maker(n):
    return lambda element : _____(a)_____ * n


def make_water(multiplier):
    nhydrogen, noxygen = 2.0, 1.0
    nmolecules = _____(d)_____
    print(str(multiplier(nhydrogen)) + ' atoms hydrogen' )
    print(str(multiplier(noxygen)) + ' atoms oxygen.' )
    print( 'Produces ' + str(multiplier(_____(b)_____)) + ' molecules of
water.' )

make_water(multiplier_maker(5.0))
```

(a) (2.0 pt) Which one of these could fill in blank (a)?

- n

SOLUTION KEY

- ~~element~~
- multiplier
- make_water
- 3.0
- 5.0

(b) (2.0 pt) Which one of these could fill in blank (b)?

- ~~nmolecules~~
- nhydrogen
- noxygen
- lambda x : x
- 10.0
- 1.0

(c) (2.0 pt) Which one of these could fill in blank (d)?

- 3.0
- 6.0
- 9.0
- ~~1.0~~
- 9.0, 3.0
- 5.0
- multiplier

SOLUTION KEY

5. (5.0 points) **Integrals and Scope**

Don't worry, you don't need to know calculus for this question! It's all about scope of variables.

```
def integral_of_eq(lower_bound):
'''given a lower_bound, integral_of_eq returns a function that gives the
definite integral of 4x^3 + 3x^2, given the upper bound'''
    def integral_with_base(lower_bound, upper_bound):
        first_value = upper_bound (a) ** 4 + upper_bound ** 3
        second_value = lower_bound ** 4 + lower_bound ** 3
        return first_value - second_value
    return lambda upper_limit: integral_with_base( lower_bound(b),
upper_limit(c) )

def lower_bound_as_1(upper_bound):
    return integral_of_eq(1)(upper_bound)

def lower_bound_as_5(upper_bound):
    return integral_of_eq(5(d))(upper_bound)

lower_bound_as_5(6)
```

(a) (1.0 pt) Which function's `upper_bound` parameter is referred to at **(a)**?

- `integral_of_eq`
- `lower_bound_as_1`
- ~~`integral_with_base`~~
- `lower_bound_as_5`
- `lambda ...`

(b) (1.0 pt) Which function's `lower_bound` parameter is referred to at **(b)**?

- `integral_with_base`
- `lower_bound_as_1`
- `lambda ...`
- ~~`integral_of_eq`~~
- `lower_bound_as_5`

(c) (1.0 pt) Which function's `upper_limit` parameter is referred to at **(c)**?

- `integral_of_eq`
- `integral_with_base`
- `lower_bound_as_1`
- `lower_bound_as_5`
- ~~`lambda ...`~~

SOLUTION KEY

(d) (1.0 pt) What should fill in blank **(d)**?

5

SOLUTION KEY