Review of Algorithms and Programming I Exam

Text: How to think like a computer scientist (learning with Python)

Reading Assignment 2: Chapter 2 (Variables, expressions and statements).

1. What's the meaning of `type (2.3)`?
2. What does `print` do?
3. What's the difference between `12 + 13` and `"12" + "13"`?
4. Consider the following Python fragment:

```
n = 3
m = 5
n = n + m
m = n - m
n = n - m
```

What values do we have in the two variables at the end?
What kind of statements are the five statements above?

Reading Assignment 3: Chapter 3 (Functions)

1. Write Python expressions that verify the following identities:
   a) $10^{\log x} = x$ for $x > 0$, and
   b) $\sqrt{3^2 + 4^2} = 5$ (Calculate both sides and print them, then look at them to compare).

2. What is a function in Python?
3. Can you define your own functions in Python?
4. What do we mean by *list of parameters* in the context of this chapter?
5. Consider the following Python program:

```
def fun(x, y):
    return x * y     # [2]

a = fun(2, 3)        # [1]
b = fun("2", 3)

print a, b
```

   a) What does it evaluate to?
   b) Replace the last statement `print a, b` with `print a + b` and explain the traceback. What's wrong?
   c) Now eliminate the line marked `[1]` and change line `[2]` to read `return x + y`. Run the program and explain the traceback.

6. Consider the following definition:

```
def fun(n, m):
    return m - n
```

Evaluate the following expressions:

a) `fun(fun(1, 2), 3)`

b) `fun(fun(1, 2), fun(3, fun(fun(4, fun(5, 6)), 7)))`

c) `fun(fun(1, 2), fun(3, fun(fun(4, fun(5, 6)), fun(7, 8))))`

d) What happens if in the definition of `fun` above we replace `return` by `print`?

7. Considering the following definitions:

```
def alpha(x, y):
    return x + beta(y, x)

def beta(x, y):
    return y - x    # [1]
```

a) What does `alpha(2, 3)` evaluate to?

b) How does the answer change if the line marked `[1]` is changed to `return x - y`?

8. Consider the following definition:

```
def fun(x):
    a = x + 1
    print a
    fun(a)
```

can you anticipate the result of calling `fun(-10)`?

Reading Assignment 4: Chapter 4 (Conditionals and recursion).

1. What's the result of calling `what(10)`?

```
def what(n):
    if n == 0:
        result = 0
    else:
        print n                      # [1]
        result = n + what(n-1)       # [2]
    return result
```

a) What's the result of calling `what(10)`?

b) Now swap the statements marked `[1]` and `[2]`: what's the result of calling (10) now?

```
def what(n):
    if n == 0:
        result = 0
    else:
        result = n + what(n-1)        # [2]
        print n                       # [1]
    return result
```

2. Consider the following two fragments:

```
if x == 5:                    if x == 5:
  x = x + 1                     x = x + 1
else:                         if x != 5:
  x = 8                         x = 8
```

Are the two fragments logically equivalent? Why or why not?