UNIX REVIEW, EXPRESSIONS, AND FUNCTIONS 1

COMPUTER SCIENCE 61A

January 24, 2013

0.1 Basic Unix Review

A quick review of basic unix functionality introduced in lab.

1. List the appropriate command to perform the following functions

   Rename a file:

   Create a new directory:

   Print the contents of a file:

   Solution: mv, mkdir, cat

2. What is the command line input to copy the file located at "~cs61a/lib/names.txt" to the current directory's parent directory?

   Solution: cp ~cs61a/lib/names.txt ..

1 Expressions

Expressions describe a computation and evaluate to a value.

1.1 Primitive Expressions

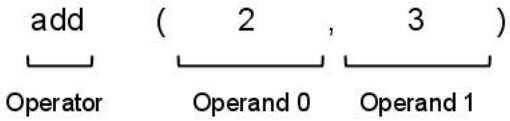
A *primitive expression* is a single evaluation step: you either look up the value of a name, or take the literal value. For example, numbers, variable names, and strings are all primitive

expressions.

```
>>> 2
2
>>> 'Hello World!'
'Hello World!'
```

1.2 Call Expressions

Call expressions are expressions that involve a call to some function. Call expressions are just another type of expression, called a *compound expression*. A call expression invokes a function, which may or may not accept arguments, and returns the function's return value. Recall the syntax of a function call:



Every call expression is required to have a set of parentheses delimiting its comma-separated operands. To evaluate a function call:

1. First evaluate the operator, and then the operands (from left to right).
2. Apply the function (the value of the operator) to the arguments (the values of the operands).

If the operands are nested function calls, apply the two steps recursively.

1.3 Questions

1. Determine the result of evaluating the following expression:

```
from operator import add, mul, sub, truediv

>>> truediv(add(mul(4, 5), sub(6, 1)), 5)
```

Solution: 5.0

2. In what order are the operators above (add, mul, sub, truediv) applied?

Solution: mul, sub, add, truediv

2 Functions

We use functions to manipulate data. Functions can be classified into two categories:

Pure function — It only produces a return value (no side effects), and always evaluates to the same result, given the same argument value(s).

Non-Pure function — It produces side effects, such as printing to the screen.

Further in the semester, we will further expand on the notion of a pure function versus a non-pure function.

2.1 Defining Functions

The structure for defining a function looks like this:

```
def <name>(<formal parameters>):
    return <expression>
```

For example, at our Python prompt we could enter the following:

```
>>> def cube(n):
...     return n * n * n
...
>>>
```

Be sure to indent the `return` statement correctly.

2.2 Questions

We have the following already defined:

```
from math import sqrt, pow

def square(x):
    return x * x
```

1. Define a function `sum_of_squares` that takes two arguments, `a` and `b`, and returns the sum of their squares.

   ```
   def sum_of_squares(a, b):
   ```

Solution:

```
def sum_of_squares(a, b):
    return square(a) + square(b)
```

2. Now define a function `distance` that takes in two sets of x-y coordinates `(x1, y1, x2, y2)` and returns the Euclidean distance between the two points.

   ```
   def distance(x1, y1, x2, y2):
   ```

Solution:

```
def distance(x1, y1, x2, y2):
    return sqrt(sum_of_squares(x1 - x2, y1 - y2))
```

The `max` function takes two numbers as argument and returns the larger of the two. For example, `max(3, 5)` returns 5.

3. Define a function `biggest_of_three` that takes three numbers, a, b, and c, and returns the largest of the three.

   ```
   def biggest_of_three(a, b, c):
   ```

Solution:

```
def biggest_of_three(a, b, c):
    return max(max(a, b), c)
```

3 Secrets to Success in CS61A

CS61A is definitely a challenge, but we all want you to learn and succeed, so here are a few tips that might help:

- Ask questions. When you encounter something you dont know, *ask*. That is what we are here for. This is not to say you should raise your hand impulsively; some usage of the brain first is preferred. You are going to see a lot of challenging stuff in this class, and you can always come to us for help.
- Go to office hours. Office hours give you time with the instructor or TAs by themselves, and you will be able to get some (nearly) one-on-one instruction to clear up confusion. You are *not* intruding; the instructors and TAs *like* to teach! Remember that, if you cannot make office hours, you can always make separate appointments with us!
- Do the readings (on time!). There is a reason why they are assigned. And it is not because we are evil; that is only partially true.
- Do (or at least attempt seriously) all the homework. We do not give many homework problems, but those we do give are challenging, time-consuming, and rewarding. The fact that homework is graded on effort does not imply that you should ignore it: it will be one of your primary sources of preparation and understanding.
- Do all the lab exercises. Most of them are simple and take no more than an hour or two. This is a great time to get acquainted with new material. If you do not finish, work on it at home, and come to office hours if you need more guidance!
- Study in groups. Again, this class is not trivial; you might feel overwhelmed going at it alone. Work with someone, either on homework, on lab, or for midterms, as long as you don't violate the cheating policy!
- Most importantly, *have fun!*