Math 480a: Sage, Spring 2011, Homework 1

William Stein

March 28, 2011

Do the following problems, and turn them in by the beginning of class on Wednesday, April 6, 2011. If you get help from a classmate or friend on any homework problem, please explicitly thank them in your write up. I have office hours Monday and Wednesday right after class in Padelford C423.

1. This problem is a warm up in using the Python programming language. Write a function called `flatten`, which takes a list and "flattens" it, that is, removes any internal lists, while keeping the elements in the same order. This is best explained by example:

```
sage: flatten([1,2,3])
[1, 2, 3]
sage: flatten(["house", [["boat"], 3]])
["house", "boat", 3]
sage: flatten([[1,2], [[[3]], 4, [[5, 6]]]])
[1, 2, 3, 4, 5, 6]
```

2. Read some papers:

   (a) Read the 1-page paper [Joyner-Stein, 2007]: `http://www.ams.org/notices/200710/tx071001279p.pdf`. What do you think?

   (b) Read as much of [Erocal-Stein, 2010] as interests you: `http://wstein.org/papers/icms/icms_2010.pdf`. What do you think?

3. In this problem, you will compare integer arithmetic using the MPIR library (as wrapped in Sage) with integer arithmetic implemented in Python itself. Please state exactly which version of Sage you are using (Sage has a `version()` command), and on what computer.

   (a) Figure out how to create a Sage integer n with 5 digits and a Python integer m with 5 digits. The following must be true about their types if you did this problem correctly:

```
sage: type(n)
<type 'sage.rings.integer.Integer'>
sage: type(m)
<type 'int'>
```

(b) What is the type of $n + m$?

(c) How many seconds are a millisecond (ms), microsecond (μs), and nanosecond (ns)?

(d) For each of the following values of d, make a d-bit Sage integer n and a d-bit Python integer m and time how long computing `n*(n+1)` and `m*(m+1)` takes. You can time evaluation of an expression as follows (type `timeit?` for more help):

```
sage: timeit('n*(n+1)')
625 loops, best of 3: 306 ns per loop
```

The values of d are: 4, 31, 64, 128, 512, 4096, 1048576.

(e) What algorithms for integer multiplication does Python implement? You might have to look at the source code of Python to determine this (maybe it is documented online). Remark: Some interesting problems in math can be encoded as "multiply these two integers"; this problem helps you see that the software you choose to do the multiplication can have an impact on how long it takes!

(f) Roughly how many nanoseconds does it take a computer to multiply two 32-bit int's using a compiled language? The following snippet of Cython code, which you can paste into the Sage notebook, could be useful in answering this question, or you could just write a little C program.

```
%cython
def f(int n):
    cdef int s=1, i
    for i in range(1,n):
        s = s*i
    return s
```

(g) Express an opinion: Why do *you* think Python doesn't implement as fast of integer multiplication as Sage? Why is it acceptable to the Python developers that multiplying some million digit integers can easily take a hundred times longer in Python than in Sage. (You might even be able to find an answer to this on the Python mailing lists...?)