Recitation 4: Cilk and Reducers

In this recitation you will gain more experience writing parallel programs using cilk by parallelizing a polynomial multiplication routine.

1 Getting started

We recommend that you work on course machines.

```
$ ssh username@cloudN.csail.mit.edu
```

To get a local copy of the repository for your work, you need to use git to clone it.

```
$ git clone /afs/csail.mit.edu/proj/courses/6.172/student-repos/fa14/recitations/
recitation4/username.git recitation4
```

2 Polynomial multiplication

In your recitation4 directory you can find a simple polynomial class that supports multiplication using two algorithms: a naive $O(n^2)$ routine, and the karatsuba algorithm which multiplies two $n$ degree polynomials in $O(n^{\log_2 3})$. See http://en.wikipedia.org/wiki/Karatsuba_algorithm for details.

The main function in main.c generates two random polynomials, multiplies them using the naive and karatsuba algorithm, and then verifies that both algorithms produce the same result.

- **Exercise:** Expose as much parallelism as you can in the karatsuba algorithm to maximize its theoretical parallelism. Compute the work and span of your parallelized program.

2.1 Burdened parallelism

Sometimes the cost of spawning small pieces of parallel work can cause your programs to run slowly. Compare your maximally parallelized version of the karatsuba routine to the serial version. You can deactivate all of the cilk keywords by including the header:

```
#include <cilk/cilk_stub.h>
```

- **Exercise:** Using any method you like, maximize the runtime of the parallel karatsuba routine on 12 cores. You should ensure that your parallel program runs faster than its serial counterpart.