C++11 Threads

Hello

```
#include <iostream>
#include <thread>

void hello() {
    std::cout << "Hello, world" << std::endl;
}

int main(int argc, char *argv[]) {
    // launch the thread
    std::thread t1(hello);

    // have main wait for t1
    t1.join();

    return 0;
}
```

Many Hellos

```
#include <iostream>
#include <thread>

void hello(int tid) {
    std::cout << "Hello from thread " << tid << std::endl;
}

int main(int argc, char *argv[]) {
     const int NUM_THREADS = 10;
    // keep track of the threads
    std::thread t[NUM_THREADS];

    // launch the threads
    for (int i = 0; i < NUM_THREADS; i++) {
        t[i] = std::thread(hello, i);
    }

    std::cout << "Hello from main" << std::endl;

    // have main wait for the hello threads
    for (int i = 0; i < NUM_THREADS; i++) {
        t[i].join();
    }
}
```

Compile Instructions

- $ g++ -std=c++11 -pthread hello.cpp -o hello

Exercises

- Remove the calls to join. What happens?
- Does the same thing happen in Java?
- Why or why not?

Variation 1/2

```
#include <iostream>
#include <thread>
#include <sstream>

void hello(int tid) {
    std::ostringstream oss;
    oss << "Hello from thread " << tid << std::endl;
    std::cout << oss.str();
}
```

Variation 2/2

```
int main(int argc, char *argv[]) {
    const int NUM_THREADS = 10;
    // keep track of the threads
    std::thread t[NUM_THREADS];

    // launch the threads
    for (int i = 0; i < NUM_THREADS; i++) {
        t[i] = std::thread(hello, i);
    }

    std::ostringstream oss;
    oss << "Hello from main" << std::endl;
    std::cout << oss.str();

    // have main wait for the hello threads
    for (int i = 0; i < NUM_THREADS; i++) {
        t[i].join();
    }

    return 0;
}
```