Assignment 7: Putting It All Together:
Register Allocation and Runtimes
Due: 5pm, Friday, April 29

1. Make a `svn` copy of your Assignment 6 solution, and copy in the contents of `src/a7`.

2. The new wrapper code takes the input `myprog.bcs` and compiles it down into assembly code in `myprog.bcs.s`.

3. The main task of this assignment is to implement an optimistic register allocator in `Color.hs`.
   A lot of support code has been provided: you do not have to build the interference graph, nor must you actually insert spill code. (However, it is “new” code this semester. If you see anything suspicious, please let me know ASAP!)
   The input to your code is the interference graph (of type `IGraph.IGraph`), an initial “coloring” (that maps machine registers to themselves), the available hardware registers, a function that estimates how bad it would be to spill any particular temporary, and a list of register-register moves. The output will be a coloring (a mapping from temporaries to machine registers) and (2) a list of temporaries that need to be spilled. If spilling is required, your coloring will be ignored; instead, the wrapper code will rewrite the code with spills and re-run your allocator.
   You also not required to implement coalescing.
   This means that the optimistic algorithm is: pick nodes/temporaries from the graph one by one, until the graph is empty; prefer to remove nodes of degree $\leq k$. Then, in reverse order, assign colors to each node. If a node cannot be colored, add it to the list of registers to spill.

4. Create a file `runtime.c` that implements the functions described in your `runtime.h`. Make sure that it compiles with `gcc -c`.

5. Finally, add to your `runtime.c` file a function `main` that performs the following tasks:
   - Calls `_initializers()` (The support code you were given handles Globals by allocating space in the data segment, and defining a function `_initializers()` that initializes all the globals.)
   - Calls `Start__main` (or whatever label your compiler generates for the static function `Start.main`), passing it any appropriate arguments.

You can then create executables by saying, e.g.,

```
gcc -o myprog -m32 myprog.bcs.s runtime.c
```

6. Provide a file `demo.bcs` that compiles and runs under your compiler. It should do something visible; preferably, a moderately interesting something.