322 Compilers Assignment: graph-test
Register allocation, graph coloring testing

Due Thursday April 19th, noon

Your job: Design test cases for the graph coloring and graph construction phases of your compiler as a pair of files, an L2 input function and the expected output. This is the shape of the coloring function:

```
graph-color : (i ...) -> ((var var ...) ...)
                         ((var reg) ...) or #f
```

The `graph-color` function accepts an L2 function (as a list of instructions), and returns two values: a graph to color and either a coloring for that graph or `#f`, indicating the algorithm failed to color the graph.

The graph should be printed out as an adjacency table, mapping each variable to its neighbors. Sort each sequence of neighbors alphabetically by the name of the variable and sort the entire graph by the names of the variables for each node. The graph should include all of the registers except `ebp` and `esp`.

The coloring should be printed out as a sequence of pairs, mapping each non-register variable to a register (color), sorted by the name of the variable. If all of the verticies in the graph printed as the first result have fewer than 5 neighbors, then the result must not be `#f`. Otherwise, it should only be `#f` when the graph cannot be colored (but the checker doesn't check this aspect).

The graph-color function should be wrapped up into a script that accepts a filename naming a file that contains the arguments in the file. The script should write their answers to stdout.

For example, if the file `f.L2f` contains:

```
((x <- 1) (eax += x) (return))
```

Then this transcript shows how your script might behave (the `ebx` in the last line might be a different register):

```
% graph-color f.L2f
((eax ebx ecx edi edx esi x)
 (ebx eax ecx edi edx esi)
 (ecx eax ebx edi edx esi)
 (edi eax ebx ecx edx esi x)
 (edx eax ebx ecx edi esi)
 (esi eax ebx ecx edi edx x)
 (x eax edi esi))
((x ebx))
```

For example, if the file `g.L2f` contains:

```
((r:x <- eax)
 (r:x += ebx) (r:x += ecx) (r:x += edx)
 (r:x += edi) (r:x += esi) (r:x += eax))
```

then this is how your script must behave:

```
% graph-color g.L2f
((eax ebx ecx edi edx esi r:x)
 (ebx eax ecx edi edx esi r:x)
 (ecx eax ebx edi edx esi r:x)
 (edi eax ebx ecx edx esi r:x)
 (edx eax ebx ecx edi esi r:x)
 (esi eax ebx ecx edi edx r:x)
 (r:x eax ebx ecx edi edx esi))
#f
```

Use `check-coloring` to make sure your colorings are sensible.

Hand in your assignment by sending email with the subject `graph-test` to `robby@eecs.northwestern.edu`. The email should include an attachment named *`name`*`.graph-test.tar.gz`. The *`name`* should be your last name in all lowercase letters unless you are pair programming, in which case it should be both last names in alphabetical order, separated by `+`. For example, if Robert Jordan and Brandon Sanderson were pair programming and handing in this assignment, they'd send in a tarfile named `jordan+sanderson.graph-test.tar.gz`.

The attachment must contain a single directory named `graph-test` containing the test cases.

The input files should use the suffix `.L2f` and the correct answers should use the suffix `.gres`. Your scripts must run on the t-lab machines (under linux).