GIMPLE IR MANIPULATION

1. The files given to you (plugin.c, Makefile) describes a dynamic plugin for an intraprocedural pass to iterate over the gimple statements, and print the global and local variables. Run it on file test.c and inspect the dump. You will observe that the same global variable list is printed each time a function is analyzed. This is because the pass is intraprocedural, and the varpool_node data structure (link list of global variables) is independent of individual functions.

   - Convert the pass into an Interprocedural pass, so that the global variable list is printed only once.
   - Convert the given dynamic plugin into the corresponding static plugin. Use '`make cc1`' to compile.
   - Modify the interprocedural plugin code to count the number of global pointers, and number of local pointers for each function.

2. Modify the dynamic plugin to implement an interprocedural pass that identifes the conditional gimple statements where either LHS or RHS is a pointer (for eg., statements like p == q where either p or q is a pointer)

3. Try to find the caller and callee relationship between the cgraph nodes. For each node, print the list of cgraph nodes that it calls. For example, for the code

```
main ()
{
   func_a ();
   func_b ();
}

foo ()
{
   func_x ();
}
```

   The output should be

```
main  -->  fun_a  func_b
foo   -->  func_x
```

NOTE

- The dump for intraprocedural tree passes can be generated by command-line switch
  `<path_to_4.6.0-install> -fdump-tree-all -O2 filename.c`
  The dump for interprocedural ipa passes an be generated by command-line switch
  `<path_to_4.6.0-install> -fdump-ipa-all -O2 filename.c`
  The dump file for the pass will have the extension '`gimple_manipulation`'.
- Parametrize the Makefile given to you according to the location of install folder (for gcc-4.6.0 pristine install) on your directory structure and the compliation flags that you would want your testcase to be compiled with.