openBeagle Notes J. Imada Feb/08

website: http://beagle.gel.ulaval.ca/

1) Installation: I installed openBeagle (v 3.0.3) locally in my subdirectory
2) Steps taken to create my own GP:
   a) Copy an example (e.g. examples/GP/symbreg) to a new directory, "myGP". There should also be a subdirectory "myGP/myGP".
   b) Revise the following files to suit your own GP
      - MyGPMain.cpp
      - MyGPEvalOp.cpp
      - MyGPEvalOp.hpp
      - You may also need to add new primitive files to define functions
   c) Revise the following files in subdirectories "myGP" and "myGP/myGP" to reflect project name "myGP" and the paths. Also, add the names of any other files (eg, new primitives) that need to be compiled.
      - Configure.ac
      - Makefile.am
      - All ".conf" files
   d) To produce a binary for "myGP":
      - ./bootstrap
        (ran this a couple of times; this creates a "configure" file from configure.ac and also makes Makefile.in)
      - ./configure
        (creates a Makefile from Makefile.in)
      - make clean
        (cleans up files not needed)
      - make
        (compiles and creates executable "myGP" in subdirectory "myGP/myGP")
   e) To run "myGP"
      ./myGP
      - This produces file: beagle.obm.gz
      - To unzip: gunzip beagle.obm.gz
      - File beagle.obm is an xml file and can be formatted if you upload it to the beagle visualizer (http://beagle.gel.ulaval.ca/visualizer/) Note: you have to create an account first.
3) ".conf" file to control GP Parameters
   - You can generate a ".conf" file to change GP parameters and detail of output by running:
     ```
     ./myGP –OBec.conf.dump=myGP.conf
     ```
   - This creates file "myGP.conf" which you can now edit
   - To run with this parameter file:
     ```
     ./myGP –Obec.conf.file=myGP.conf
     ```
4) To make Strongly-typed:

- Revise ".conf" file to change to "constrained" classes (see user's manual)

5) To minimize fitness:
    1) Revise MyGPMain.cpp (around Vivarium instantiation)
    2) Revise MyGPEvalOp.cpp (to return FitnessSimpleMin in evaluate method)
    3) Here's my code…

```
// 2: Build a system.
   GP::System::Handle lSystem = new GP::System(lSet);
//***added by J. Imada to minimize fitness
//    lSystem->getFactory().setConcept("Fitness", "FitnessSimpleMin");
//end of add
   // 3: Build evaluation operator.
   SimpleGPEvalOp::Handle lEvalOp = new SimpleGPEvalOp;
   // 4: Build an evolver and a vivarium.
   GP::Evolver::Handle lEvolver = new GP::Evolver(lEvalOp);
//***added by J. Imada to minimize fitness
   GP::Tree::Alloc::Handle lTreeAlloc = new GP::Tree::Alloc;
   FitnessSimpleMin::Alloc::Handle lFitAlloc = new FitnessSimpleMin::Alloc;
//end of add
//*** revised (J. Imada) constructor in the following line
   GP::Vivarium::Handle lVivarium = new GP::Vivarium(lTreeAlloc, lFitAlloc);
```

6) To add "C" functions and other external programs
    - Add {extern} to C header files
    - Revise the Makefile in (myGP/myGP) before the "make" to reflect paths for the external programs
        LDFLAGS (for –L's)
        CPPFLAGS (for –I's)
        LIBS (for –l's)
    - Also may need to change environment variable LD_LIBRARY_PATH in .bash_profile file