Using the Grammar Development Environment (GDE)

1. Login to johann

2. Invoke Lisp

   ```
   $ kcl
   ```

3. Load the GDE

   ```
   > (load "/usr/course/3/NLP/GDE")
   ```

4. Run the GDE

   ```
   > (gde-top-loop)
   ```

5. There are 3 modes within the GDE. To begin with you will be in development mode. The prompt is

   ```
   Gde >
   ```

   The following commands will be useful

read filename	--reads in a grammar from filename. The example grammars I supply will be in /usr/course/3/NLP/gram1, etc.
cl	--clears the GDE (do this prior to reading in a new grammar)
q	--quit (it will check with you (twice?) before allowing you to quit)
p	--enter parser mode
g S	--enter generator mode (you can also enter it with 'gen NP' 'gen VP' or whatever you want it to generate).

   After issuing q (quit), and confirming that you want to quit, you will be back in Lisp. To leave Lisp:

   ```
   > (bye)
   ```

   Now you will be back to Unix.

6. Parser mode, entered from development mode using p, should only be entered after reading in a grammar. The prompt is

   ```
   Parse >>
   ```

   Type a sentence and it will tell you how many parses it has.
   Other useful commands

help	--obvious
v l	--after parsing, to show the parses in labelled bracket form
v f	--after parsing, to show the syntax tree
q	--to return to development mode

7. Generator mode, entered from development mode using g S (or whatever). The prompt is

   ```
   Gen >>
   ```

   Useful commands

help	--obvious
a b 2	--generate some 'random' sentences of length two (or whatever)
q	--return to development mode

Using the Grammar Development Environment (GDE)
****** NOW WITH ADDED SEMANTICS ******

1. To begin with things are pretty much as before:
   Login to johann, invoke Lisp (kcl) and load the GDE. However, to load the version of the GDE which has semantic interpretation facilities, instead of (load "/usr/course/3/NLP/GDE"), you type (load "/usr/course/3/NLP/GDE2").
   Now run the GDE (gde-top-loop).
   Everything is much as before, except that there are now more options in parse mode, these options being the ones that allow you to see the semantics.

2. gram5 is nearly the same as gram1 (there are one or two simplifications), but it has semantics attached to the rules and lexical entries.
   Here are some of the new options which you can try with gram5 once you are in parse mode:

help	--obvious
v l	--after parsing, to show the parses in labelled bracket form
v full	--after parsing, to show the syntax tree
v s	--after parsing, to see the semantics of the sentence
v s u	--ditto, for semantics which is 'unreduced' (no lambda conversion done)
v form	--after parsing, to see the semantics associated with a node in the tree (you will be asked which node's semantics you want to see).
v form u	--ditto, for 'unreduced' semantics
q	--to return to development mode

3. That was GDE2. There is yet another version of the GDE, GDE3.
   You run things as before but this time you load GDE3 instead of GDE2.
   This loads a logic database facility.

4. With GDE3 you should read in gram6. This is a quite extensive grammar with semantics and is of a complexity greater than is covered in the course.

   Now in parse mode you have another option. After parsing a sentence (and looking at its semantics) you can issue the command int (= interpret). What this does is to convert your semantic representation into the logic database notation. If your sentence was declarative, the database will be updated with the semantic representation; if your sentence was a wh-question or a yn-question, then the semantic representation will be used as a query to the database and an answer will be returned. Thus you can do something like the following:

   >> Macbeth stabs Duncan

   >> int

   >> does Macbeth stab Duncan

   >> int

   and the system responds "Yes".

   >> who stabs Duncan

   >> int

   and the system responds "Macbeth".

   etc.