I included the translations of each of the sentences for exercises 2 and 3 for numbers 1 through 4 in this assignment. So as to check the well-formedness of these formulae, set up the Expression.fromstring() in the following fashion:

```
>>> import nltk
>>> check = nltk.sem.Expression.from_string
```

Exercise 8.2

1. Angus likes Cyril and Irene hates Cyril.
   Translation: like(Agnus, Cyril) & hate(Irene, Cyril)
   Description: 'like(Agnus, Cyril)' translates to 'Angus likes Cyril' and 'hate(Irene, Cyril)' translates to 'Irene hates Cyril'.
   Code Snippet:

   ```
   >>> check('like(Agnus, Cyril) & hate(Irene, Cyril)')
   <AndExpression (like(Angus, Cyril) & hate(Irene, Cyril))>
   ```

2. Tofu is taller than Bertie.
   Translation: taller(Tofu, Bertie)
   Description: 'taller(Tofu, Bertie)' translates to 'Tofu is taller than Bernie'.
   Code Snippet:

   ```
   >>> check('taller(Tofu, Bertie)')
   <ApplicationExpression taller(Tofu, Bertie)>
   ```

3. Bruce loves himself and Pat does too.
   Translation: love(Bruce, Bruce) & love(Pat, Pat)
   Description: 'love(Bruce, Bruce)' translates to 'Bruce loves Bruce, i.e, himself' and 'love(Pat, Pat)' translates to 'Pat loves Pat, i.e, himself'.
   Code Snippet:

   ```
   >>> check('love(Bruce, Bruce) & love(Pat, Pat)')
   <AndExpression (love(Bruce, Bruce) & love(Pat, Pat)>
   ```

4. Cyril saw Bertie, but Angus didn't.
   Translation: see(Cyril, Bertie) & see(Angus, Bertie)
   Description: 'see(Cyril, Bertie)' translates to 'Cyril sees Bertie' and ' see(Angus, Bertie)' translates to 'Angus does not see Bertie'.
   Code Snippet:

   ```
   >>> check('see(Cyril, Bertie) & ~see(Angus, Bertie)')
   <AndExpression (see(Cyril, Bertie) & ~see(Angus, Bertie))>
   ```

Exercise 8.3

1. Angus likes someone and someone likes Julia.
   Translation: $\exists x$ like(Angus, x) & $\exists y$ like(y, Julia)
   Code Snippet:

```
>>> check('exists x.like(Angus, x) & exists y.like(y, Julia)')
<AndExpression (exists x.like(Angus, x) & exists y.like(y, Julia)>
```

2. Angus loves a dog who loves him.
   Translation: $\exists x$ (dog(x) & love(Angus, x) & love(x, Angus))
   Code Snippet:

```
>>> check('exists x.(dog(x) & love(Angus, x) & love(x, Angus))')
<ExistsExpression exists x.(dog(x) & love(Angus, x) & love(x, Angus))>
```

3. Nobody smiles at Pat.
   Translation: $\exists x$ (smile(x, Pat))
   Code Snippet:

```
>>> check('~exists x.smile(x, Pat)')
<NegatedExpression ~exists x.smile(x, Pat)>
```

4. Somebody coughs and sneezes.
   Translation: $\exists x$ (cough(x) & sneeze(x))
   Code Snippet:

```
>>> check('exists x.(cough(x) & sneeze(x))')
<ExistsExpression exists x. (cough(x) & sneeze(x))>
```