This assignment is about natural language generation. It uses the FUF/SURGE natural language generation environment. Make sure to go through all appropriate documentation before you start working. The resources page on the course Web page includes two important manuals: fuf5.2.ps and fuf-short-manual.ps - consult these when needed - you won’t need to read the entire documents. The FUF web site http://www.cs.bgu.ac.il/surge/index.html has more information, including a description of SURGE.

Some portions of this assignment use the SURGE grammar and others ask for your own grammar. Please read the problem statements carefully to avoid confusion.

IMPORTANT: Start early as this is a long assignment.
1 (10 points) Getting started with Allegro and FUF.

The first exercise teaches you the basic format of a FUF grammar and of a (semantic) input FD.

Type the following two commands:

cd /data0/temp/fufsurge
/data0/tools/acl82express/alisp

Now you are inside Allegro.

Run the following three commands:

(load "server")
(in-package "FUG5")
(setq *default-pathname-defaults* #p"/data0/class/nlpw11/hw3/"

(load "example1.l")

Submit the output of the following commands:

(get-test 'i1)
(pprint (get-test 'i1))
(pprint (example1))
(uni (get-test 'i1))
(trace-on) (uni (get-test 'i1)) (trace-off)

Annotate the output of these commands to show that you understand what each line means. For the last command, please make sure to show that you understand how unification is performed.

Note: additional FUF examples are available in /data2/tools/fuf - it is definitely worth exploring these directories.
2 (50 points) Writing grammars using FUF

Now you will have to make some modifications to your grammar. First, copy the current grammar from /data0/class/nlpw11/hw3/example1.l to your own directory and save it as example2.l. Note that you will now have to give a different value to *default-pathname-defaults* so that you can access your own grammar. Don’t forget the trailing slash at the end of the directory name! You will be making changes to example2.1. Submit the new version of the grammar as well as all your input tests and the output of FUF.

In this part of the assignment, you will use the FUF system to build a simple natural language generator. More specifically, you will use as input a set of database records and produce a sentence for each of the records.

The input should consist of one item of each of the following semantic attributes: feeder, fed, utensil, food, time. Figure ?? lists all possible values that each attribute can take.

IMPORTANT: Do NOT use SURGE for this exercise. You have to write your own FUF grammar.

<table>
<thead>
<tr>
<th>FEEDER</th>
<th>FED</th>
<th>UTENSIL</th>
<th>FOOD</th>
<th>TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mary (feminine, singular)</td>
<td>Mary (feminine)</td>
<td>silver spoon</td>
<td>fish</td>
<td>every day</td>
</tr>
<tr>
<td>Joe (masculine, singular)</td>
<td>cat (masculine/feminine, singular)</td>
<td>spoon</td>
<td>milk</td>
<td>often</td>
</tr>
<tr>
<td>they (plural)</td>
<td>cats (masculine/feminine, plural)</td>
<td>eyedropper</td>
<td>sushi</td>
<td>today</td>
</tr>
<tr>
<td>I (masculine/feminine, singular)</td>
<td>Joe (masculine)</td>
<td>- (none)</td>
<td>dog food</td>
<td>now</td>
</tr>
<tr>
<td>- (none)</td>
<td>fish (singular)</td>
<td>- (none)</td>
<td>- (none)</td>
<td></td>
</tr>
<tr>
<td>- (none)</td>
<td>fish (plural)</td>
<td>- (none)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- (none)</td>
<td>dog (singular)</td>
<td>- (none)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- (none)</td>
<td>dogs (plural)</td>
<td>- (none)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 1: Possible values for the five input slots.

Here is a sample input. Note that I am using the word “value” instead of “lex”. This distinction is very important. The grammar in the first example could simply plug in the lexical representation of a word into its output. In this exercise, you have to process the value and convert it to a (possibly different) lexical word - e.g., if both feeder and fed have the value “Mary”, you will have to translate the second word to “herself”. See Figure ?? for an example.

```
(def-test i2
  "Every day, Mary feeds her cat with a spoon."
  ((cat clause)
   (tense present)
   (feeder ((value "Mary")
     (gender feminine)))
   (fed ((value "cat")
     (number singular)))
   (time ((value "every day")))
   (utensil ((value "spoon")))))
```

Figure 2: Sample input for Question 2. Note that some grammatical properties (e.g., gender and number) cannot be inferred from the lexical value alone so they have to be specified in the input.

Submit ten input FDs and the corresponding output from FUF. Pick your examples to be
representative of the input specification and show that your grammar can handle the linguistic phenomena below:

(a) simple sentences (e.g., “John feeds Mary.”) : 10 points
(b) possessive (“her”) : 5 points
(c) agreement (“Mary” - “her dog”) : 5 points
(d) reflexives (“Mary” - “herself”) : 5 points
(e) proper use of articles (e.g., avoiding “the Mary”) : 5 points
(f) time expressions (“today”) : 5 points
(g) prepositional phrases (“with a spoon”) : 5 points
(h) passive voice (“the dog is fed”) : 5 points
(i) reasonable word order (e.g., avoiding “John feeds every the dog day”) : 5 points
3 (40 points) SURGE

For this exercise, you will be using SURGE, a large grammar of English written in FUF.

To start it, use the following commands:

```
(load "server")
(in-package "FUG5")
(setq *default-pathname-defaults* #p"/data2/tools/surge/inputs/")
(load "ir.l")
(test :item 't1)
```

Submit the input for tests t1, t27bis, t362, and t363, as well as the corresponding output of SURGE. You can use variations of the following command:

```
(uni-string (get-test 't1))
```

Now, look at the circumstantial file `circum.l`, in the same directory as `ir.l`.

```
(load "circum.l")
```

Using SURGE, generate the following sentences:

- Meow is a cat.
- Every day, Joe feeds his dog with a silver spoon.
- My friend Jennifer lives in Spain.
- My friend Jason doesn't live in Spain.
- Mary has a little lamb and Jack has a beanstalk.
- The police announced that the thief was captured.
- The University of Michigan is located in Ann Arbor, Michigan.
- Even if available tonight, Mark won't come to the meeting.
- Firefighters are seeking to restrain the Old Fire approaching Lake Arrowhead, a resort community in the San Bernardino Mountains east of Los Angeles.
- Online travel agency Orbitz has notified law enforcement authorities about a recent security breach that has allowed spammers to obtain customers’ e-mail addresses.

For each sentence, submit the input FD and the string produced by SURGE. You should not have to modify SURGE for this exercise however don’t expect it to handle all possible sentences. If you cannot quite get the expected output, try to be as close to it as possible while keeping the sentence grammatical. You will get partial credit for partial solutions.
**Some useful commands**

<table>
<thead>
<tr>
<th>ALLEGRO COMMANDS:</th>
</tr>
</thead>
<tbody>
<tr>
<td>(quit) - leave Allegro</td>
</tr>
<tr>
<td>:exit - leave Allegro</td>
</tr>
<tr>
<td>:pop - return to previous level in case of error</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FUF COMMANDS (assuming i1 is defined via def-test):</th>
</tr>
</thead>
<tbody>
<tr>
<td>(pprint FD) - pretty prints a given FD</td>
</tr>
<tr>
<td>(test :item 'i1) - unifies test i1 with the current grammar and prints the generated string. It also indicates if the generated string is the expected one or not.</td>
</tr>
<tr>
<td>(get-test 'i1) - returns the FD part of a test (ignoring the expected output).</td>
</tr>
<tr>
<td>(uni-fd (get-test 'i1)) - produces the output FD of the unification of test i1 with the current grammar.</td>
</tr>
<tr>
<td>(uni-string (get-test 'i1)) - produces the output string of the unification of test i1 with the current grammar.</td>
</tr>
</tbody>
</table>

**Credits:** FUF and Surge were written by Michael Elhadad, now at Ben-Gurion University in Israel.