CS 335 - Homework 7

Deadline:

Due by 11:59 pm on Friday, April 15th

How to submit:

Submit your files for this homework using `~st10/335submit` on nrs-labs, with a homework number of 7

Purpose:

To read and think about automatic storage reclamation, manual storage reclamation, and garbage collection, and to complete development of your Prolog game.

Important notes:

- Note that this homework's responses for problems 1-3 will eventually be gathered and made available to all class members. So, please indicate if you would like for your name to be included with your responses or not in this eventual compilation.
- You are expected to use SWI-Prolog for the Prolog game portion of this assignment.

The Problems:

Aside 1:

Read MacLennan, Section 11.2, Storage Reclamation, pp. 388-394.

Then read the following dandy introductory article on garbage collection in Java:

Brian Goetz's "Java Theory and Practice: A Brief History of Garbage Collection"

...available at:

http://www.ibm.com/developerworks/java/library/j-jtp10283/

(This paper includes links to more references to automatic storage reclamation and garbage collection, including two follow-up articles by the same author in the same series discussing how later versions of Java actually do garbage collection.)

Problem 1 - 12 points

Give a list of at least two advantages of manual storage reclamation (MacLennan calls this explicit erasure), and give a list at least two disadvantages of manual storage reclamation. Then note which of these you find most compelling, and why.

Problem 2 - 12 points

Now give a list of at least two advantages of automatic storage reclamation (MacLennan calls this automatic erasure), and give a list of at least two disadvantages of automatic storage reclamation. And, again, then note which of these you find most compelling, and why.

Problem 3 - 12 points

Finally, based on the Brian Goetz paper, write a paragraph (at least 3 sentences) describing at least one thing that either surprised you or interested you (your choice) related to garbage collection from this article, and why.

Aside 2:

Here is an interesting predicate provided by SWI-Prolog: `random/1`

Interestingly, you cannot get its listing using `listing/1` (it is listed as being "foreign). However, in the SWI-Prolog documentation at: http://www.swi-prolog.org/pldoc/man?predicate=random%2F1

...it states:

"random(+IntExpr)

Evaluates to a random integer i for which 0 =< i < IntExpr. The system has two implementations. If it is compiled with support for unbounded arithmetic (default) it uses the GMP-library random functions. In this case, each thread keeps its own random state. The default algorithm is the Mersenne Twister algorithm. The seed is set when the first random number in a thread is generated. If available, it is set from /dev/random. Otherwise it is set from the system clock. If unbounded arithmetic is not supported, random numbers are shared between threads and the seed is initialised from the clock when SWI-Prolog was started. The predicate set_random/1 can be used to control the random number generator."

So, here is a sampling from an example swipl session playing with this predicate:

```
?- X is random(100).
X = 34.

?- X is random(100).
X = 9.

?- X is random(100).
X = 70.

?- X is random(100).
X = 88.

?- X is random(100).
X = 41.

?- X is random(100).
X = 15.

?- X is random(100).
```

```
X = 13.

?- X is random(100).
X = 9.

?- X is random(100).
X = 98.

?- X is random(100).
X = 75.
```

Problem 4 - 64 pts

Complete your Prolog adventure game that you have been working on in Homeworks 5 and 6, with one additional requirement:

- you must somehow use `random/1` in your game

(There are also a few additional writing-pieces you need to include for this final version -- please read these specifications carefully!)

By this homework's (Homework 7's) deadline, you will be expected to submit working attempts at all **four** of the requirements locked-door/hidden-object/incomplete-object/limited-resources, and you must have implemented at least one list, at least some use of arithmetic, and at least one use of `random/1` in your game.

You need to submit (for Homework 7):

- your knowledge base thus-far (in a file named `hw7-game3.pl`),
- a transcript of a sample run of this latest version of your game (in a file named `hw7-game3-sample.txt` created by the `protocol` predicate)
  - (you may include more than one sample run transcript if you like -- if you do, name them `hw7-game3-sample1.txt`, `hw7-game3-sample2.txt`, etc.
- a `hw7-game3-readme.txt` file that:
  - briefly describes your game,
  - briefly describes EACH of your locked-door/hidden-object/incomplete-object/limited-resource,
  - briefly describes how you are using lists in your game,
  - briefly describes how you are using arithmetic in your game,
  - briefly describes how you are using `random/1` in your game,
  - states whether or not you are using cuts (!) in your game, and if so, briefly describes how you are using them,
  - briefly describes your favorite aspect of your game.

You are expected to turn in a **working** version of your game, meeting **all** of the above specifications, by this homework's deadline. Your Homework 7 grade will based on whether it does so, at that point.

HOWEVER -- be aware that you will be demonstrating your game in class at some point. (I will let you know when I have determined the date.)