Ονοματεπώνυμο: Χαριτούδης Απόστολος-Ανδρέας

Μάθημα: Λογικός Προγραμματισμός

Α.Μ.: π17178

Εξάμηνο: 7

Θέμα Εργασίας:

[Για φοιτητές με επώνυμο από Ρ έως Ω]: Αναπτύξτε πρόγραμμα adventure game που να επιτρέπει σε κάποιον να βγει από το αυτοκίνητό του, να διασχίσει το δρόμο και να μπει σε ένα βενζινάδικο από το οποίο μπορεί να αγοράσει αναψυκτικό και εφημερίδα, και στη συνέχεια να επιστρέψει στο αυτοκίνητό του

Πανεπιστήμιο Πειραιά (ΠΑ.ΠΕΙ.)

Πηγαίος Κώδικας:

```
/* Gas Station.
   Consult this file and issue the command:   start.  */

:- dynamic at/2, i_am_at/1.   /* Needed by SWI-Prolog. */
:- retractall(at(_, _)), retractall(i_am_at(_)).

/* This defines my current location. */

i_am_at(car).


/* These facts describe how the rooms are connected. */
path( car, out, road).
path(road, cross, gas_entrance).
path(gas_entrance, gin, gas_station).
path(gas_station, gout, gas_entrance):- at(soda, in_hand).
path(gas_station, gout, gas_entrance) :-
                        write('don’t forget to take the listed
item'), nl,
        !, fail.


path(gas_entrance, crossag, road).

path(road, in, car).



/* These facts tell where the various objects in the game
   are located. */

at(soda, gas_station).



/* These rules describe how to pick up an object. */

take(X) :-
        at(X, in_hand),
        write('You''re already holding it!'),
        nl, !.

take(X) :-
        i_am_at(Place),
        at(X, Place),
        retract(at(X, Place)),
        assert(at(X, in_hand)),
        write('Thank you for your purchase.'),
        nl, !.

take(_) :-
        write('I don''t see it here.'),
        nl.
```

```
/* These rules define the six direction letters as calls to go/1.
*/

out  :- go(out).

cross :- go(cross).

gin :- go(gin).

gout :- go(gout).

crossag :- go(crossag).

in :- go(in).


/* This rule tells how to move in a given direction. */

go(Direction) :-
        i_am_at(Here),
        path(Here, Direction, There),
        retract(i_am_at(Here)),
        assert(i_am_at(There)),
        look, !.

go(_) :-
        write('You can''t go that way.').


/* This rule tells how to look around you. */

look :-
        i_am_at(Place),
        describe(Place),
        nl,
        notice_objects_at(Place),
        nl.


/* These rules set up a loop to mention all the objects
   in your vicinity. */

notice_objects_at(Place) :-
        at(X, Place),
        write('There is a '), write(X), write(' here.'), nl,
        fail.

notice_objects_at(_).



/* Under UNIX, the   halt.  command quits Prolog but does not
   remove the output window. On Windows, however, the window
   disappears before the final output can be seen. Hence this
   routine requests the user to perform the final  halt.  */
```

```
finish :-
        nl,
        write('The game is over. Please enter the   halt.
command.'),
        nl, !.


/* This rule just writes out game instructions. */

instructions :-
        nl,
        write('Enter commands using standard Prolog syntax.'),
nl,
        write('Available commands are:'), nl,
        write('start.                    -- to start the game.'),
nl,
        write('out.cross.gin.gout.crossag.in.   -- to go
there.'), nl,
        write('take(Object).             -- to pick up an
object.'), nl,          write('open.                      -- to
open doors.'), nl,
        write('look.                     -- to look around you
again.'), nl,
        write('instructions.           -- to see this message
again.'), nl,
        write('halt.                     -- to end the game and
quit.'), nl,
        nl.


/* This rule prints out instructions and tells where you are. */

start :-
        instructions,
        look.


/* These rules describe the various rooms.  Depending on
   circumstances, a room may have more than one description. */

describe(car) :-
        at(soda, in_hand),
        write('Congratulations!!  You  didn't forget the can'),
nl,
        write('  of soda and won the game.'), nl,
        finish, !.

describe(car) :-
        write('You are in a car.  Cross the road'), nl,
        write('enter the gas station;'), nl,
        write('take a can of soda '), nl,
        write('rand come back in'), nl,
        write('this car.'), nl.

describe(road) :-
        at(soda, in_hand),
        write('Aaah  time to get in the car '), nl,
```

```
        write('and drive away'), nl.


describe(road) :-
        write('You are on the road cross it.'), nl,
        write('And go towards the gas station'), nl.


describe(gas_entrance) :-
        at(soda, in_hand),
        write('Let's go.'), nl.



describe(gas_entrance) :-
        write('You are here, not many people are inside just'),
nl,
        write('open the door.  Don't waste time!'), nl.

describe(gas_station) :-
        at(soda, in_hand),
        write('You got what we needed lets go,'), nl,
        write(' that's was fast. '), nl.



describe(gas_station) :-
        write('Oh look there is the shelve with  drinks.'), nl.
```

Επεξηγήσεις:

Όπως φαίνεται παραπάνω, το παιχνίδι ξεκινάει όταν πατήσουμε το start., το οποίο περιέχει τις εντολές instructions., look., οι οποίες παρουσιάζουν ποιες εντολές θα πρέπει να χρησιμοποιήσουμε για να ανατρέξουμε το πρόγραμμα επιτυχώς και μας δείχνει την τωρινή θέση (θέση έναρξης του παιχνιδιού) και δίνει περιγραφή αυτής της θέσης-χώρου αντίστοιχα. Στη συνέχεια, όταν ανατρέχεται η εντολή instructions, όπως φαίνεται στον κώδικα θα μας δείξει ποιες εντολές μπορούμε να χρησιμοποιήσουμε. Και αυτές είναι:

- Εντολές κίνησης: out., cross., gin., gout., crossag., in.,
- Εντολή αλληλεπίδρασης με τα αντικείμενα: take(Object).,

- Εντολή για άνοιγμα πόρτας: open.,
- Εντολή τερματισμού: halt., και τέλος
- Οι εντολές start., instructions., look., τις οποίες αναφέραμε προηγουμένως.

Αφού μιλήσαμε για τι εντολές που έχουμε στην διάθεσή μας, ας προχωρήσουμε στον τρόπο υλοποίησής τους. Όπως παρατηρούμε στην αρχή του κώδικα ορίζουμε δύο dynamic μεταβλητές, την at και την i_am_at, τις οποίες θα χρησιμοποιήσουμε για να δώσουμε τοποθεσία στο αντικείμενο και για να γνωρίζουμε κάθε στιγμή την τωρινή μας θέση. Φυσικά, αυτές οι μεταβλητές κάθε φορά και κυρίως η i_am_at θα αλλάζουν. Αυτό επιτυγχάνεται με το retractcall, το οποίο χρησιμοποιείται αργότερα για να μπορούμε να αλλάζουμε με επιτυχία τα ορίσματα. Κάτω από αυτές τις δύο δηλώσεις, ορίζουμε και την αρχική θέση, η οποία είναι το car και το καταφέρνουμε αυτό με το i_am_at(car). . Στη συνέχεια, ορίζουμε με ποιον τρόπο συνδέονται οι χώροι μεταξύ τους. Όπως φαίνεται μέσα στις path(), βάζουμε χώρο1, κατεύθυνση κίνησης, ώστε κάθε συνδυασμός να έχει διαφορετική κίνηση και χώρο2. Μερικοί συνδυασμοί έχουν έναν επιπλέον περιορισμό μετά την παρένθεση, για να μην πέσει σε ατέρμονη επανάληψη αν δεν πραγματοποιηθεί η εντολή take. Έπειτα, στην at ορίζουμε το αντικείμενο με το οποίο θα αλληλεπιδράσουμε και τον χώρο στον οποίο βρίσκεται. Παρακάτω, ορίζουμε την εντολή take, την οποία την χωρίζουμε σε τρεις εκδοχές. Η πρώτη είναι όταν προσπαθούμε να πάρουμε το αντικείμενο, ενώ το έχουμε ήδη στο χέρι, η δεύτερη είναι όταν δεν έχουμε το αντικείμενο στο χέρι και η τρίτη είναι όταν δεν υπάρχει το αντικείμενο που ζητάμε.

- Στην πρώτη take, ελέγχουμε αν το αντικείμενο X είναι στο χέρι και αν είναι βγάζει το απαραίτητο μήνυμα, αλλιώς αν δεν ισχύει αυτή η περίπτωση πάμε στην επόμενη.
- Στην δεύτερη take, εφόσον η πρώτη απέτυχε κοιτάμε το που βρισκόμαστε τώρα και αν το αντικείμενο X υπάρχει στην τωρινή μας θέση. Αν υπάρχει κάνουμε retract και μετά assert στην at, ώστε η at να δείχνει το αντικείμενο X

και το status in_hand και μας ειδοποιεί με το απαραίτητο μήνυμα. Έαν τώρα αποτύχει και αυτή, πάμε στην

- Στην τρίτη take, η οποία μας ειδοποιεί οτι δεν υπάρχει το αντικείμενο που αναζητούμε σε αυτόν τον χώρο.

Προχωρώντας, ορίζουμε τις εντολές κίνησης: κατεύθυνση:- go(κατεύθυνση). Οι κατευθύνσεις είναι έξι: out, cross, gin, gout, crossag, in, οι οποίες ταυτόχρονα δείχνουν με ποιους χώρους συνδέονται, οπότε κάθε κατεύθυνση είναι μοναδική για τον συνδυασμό της. Αφού ορίσαμε τις εντολές αυτές, θα δείξουμε πως λειτουργεί η go, η οποία ενεργοποιείται στο παρασκήνιο με κάθε εντολή κίνησης. Όπως φαίνεται στον κώδικα, η go παίρνει την κατεύθυνση, για παράδειγμα την gin, κοιτάει την τωρινή θέση, ελέγχει αν υπάρχει το path που να αντιστοιχεί στην τωρινή μας θέση, στην κατεύθυνση. Αν υπάρχει το path αυτό, θα κάνουμε retract την τωρινή μας θέση από το i_am_at και θα κάνουμε assert την there, που αντιστοιχεί στο path που ελέγξαμε πριν.

Και τέλος, θα καλέσουμε look, για να μας δώσει την περιγραφή της καινούριας μας θέσης. Αν δεν επιτευχθεί όλο αυτό, πάμε στην δεύτερη εναλλακτική της go, που θα μας ενημερώσει με ένα μήνυμα για την αποτυχία της κίνησης.

Αφού εξηγήσαμε όλα τα προηγούμενα, ήρθε η ώρα να εξηγήσουμε πως λειτουργεί η look. Η look αρχικά, κοιτάει την τωρινή μας θέση και καλεί μια background συνάρτηση, την describe, την οποία θα εξηγήσουμε αργότερα. Μετά την describe, θα ζητήσουμε από το πρόγραμμα να καλέσει την notice_objects_at, για να μας ενημερώσει αν υπάρχουν αντικείμενα γύρω μας. Αφού ορίσαμε την look, ώρα να ορίσουμε και την notice_objects_at.

- Στην πρώτη εκδοχή, παίρνει την θέση μας, ελέγχει αν υπάρχει αντικείμενο X σε αυτή την θέση μέσω της at(X, Place) και μας ενημερώνει με το κατάλληλο μήνυμα. Αν δεν υπάρχει, πάει

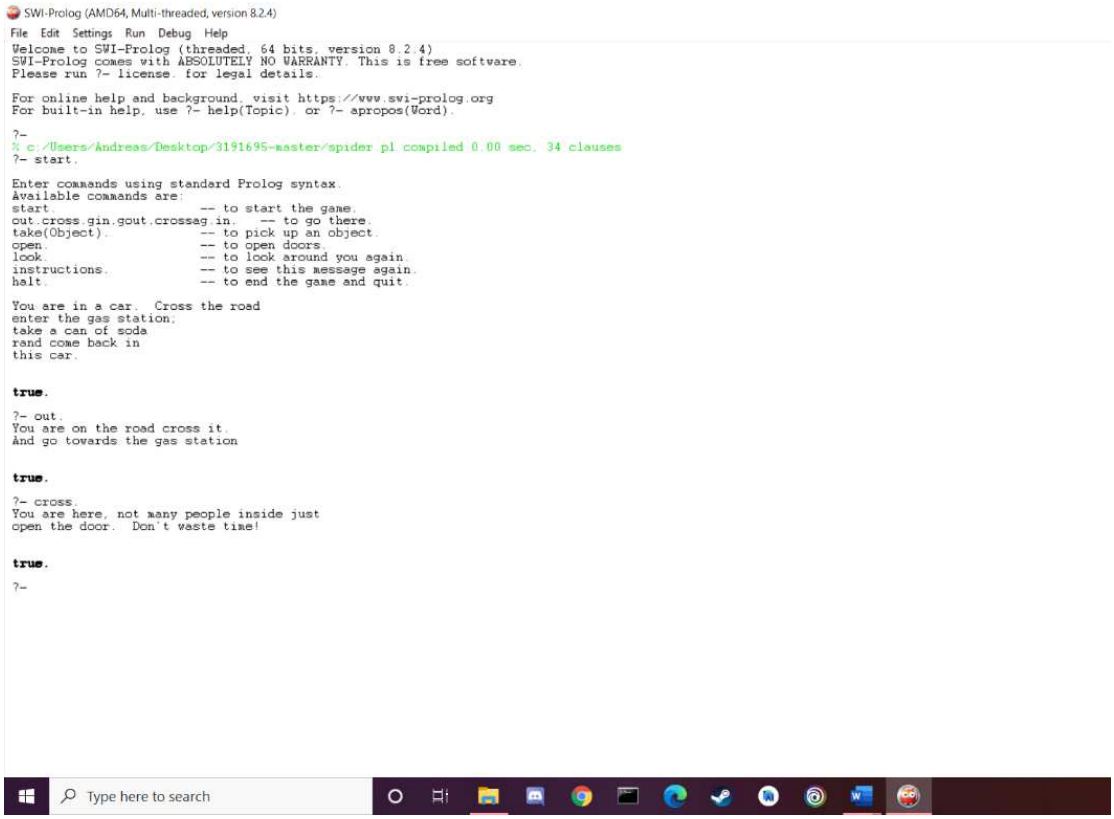
- Στην δεύτερη εκδοχή, η οποία δεν κάνει τίποτα.

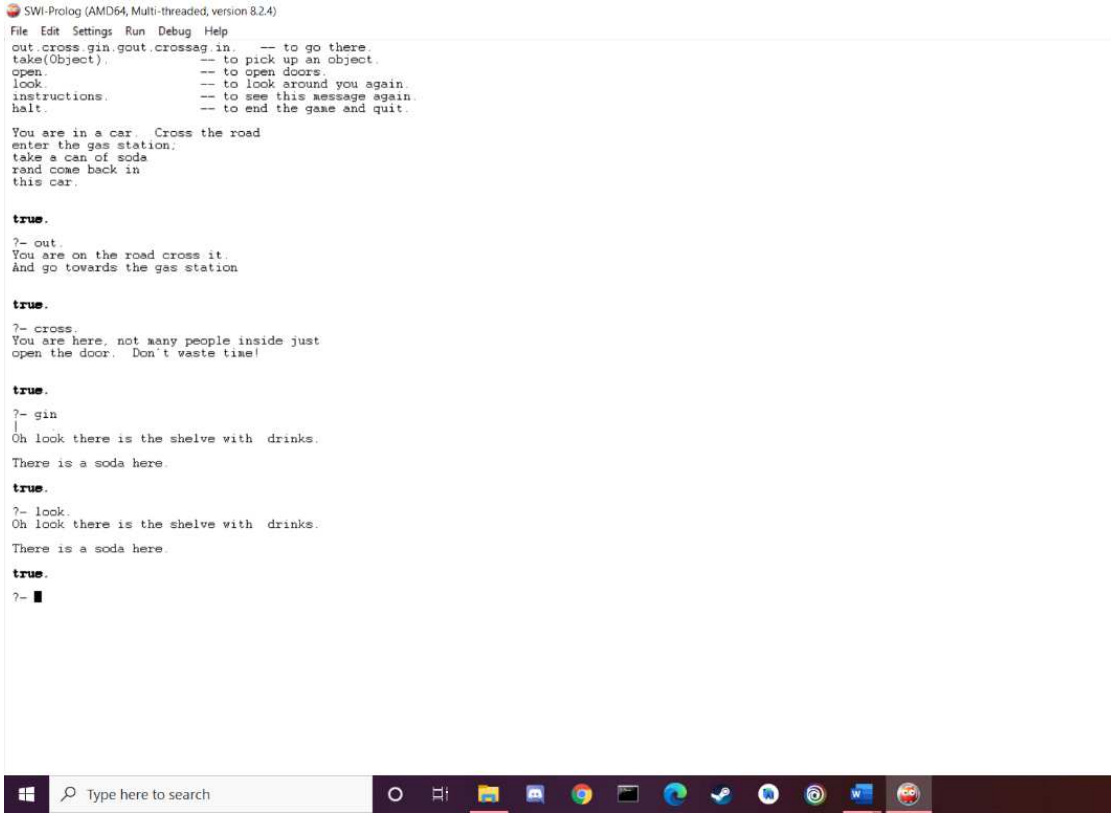
Όπως εξηγείται και στον κώδικα, ζητάμε από τον χρήστη να γράψει ο ίδιος την εντολή halt, καθώς η αυτοματοποιημένη χρήση της στα Windows δεν θα μας δείξει

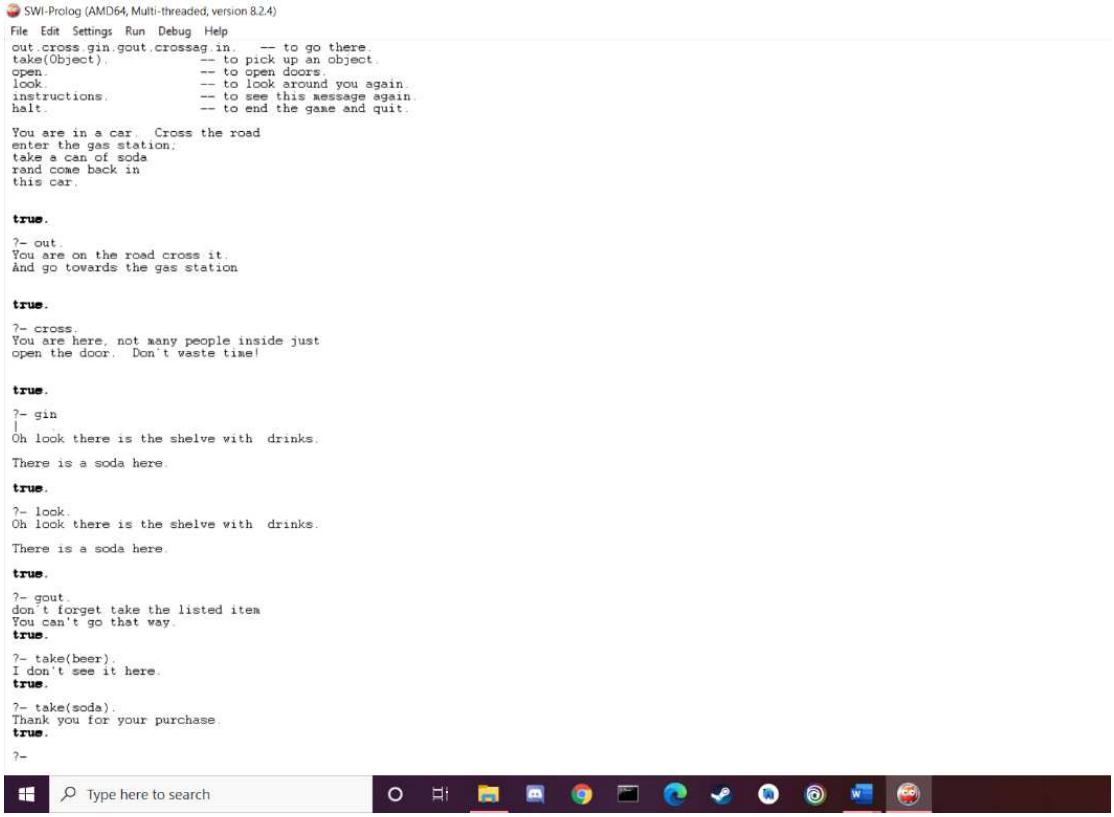
τα τελευταία μηνύματα. Για αυτό, δημιουργήσαμε την συνάρτηση finish, ώστε να ενημερώνουμε τον παίκτη οτι έχει φτάσει στο τέλος και οτι μπορεί να τερματίσει το πρόγραμμα.

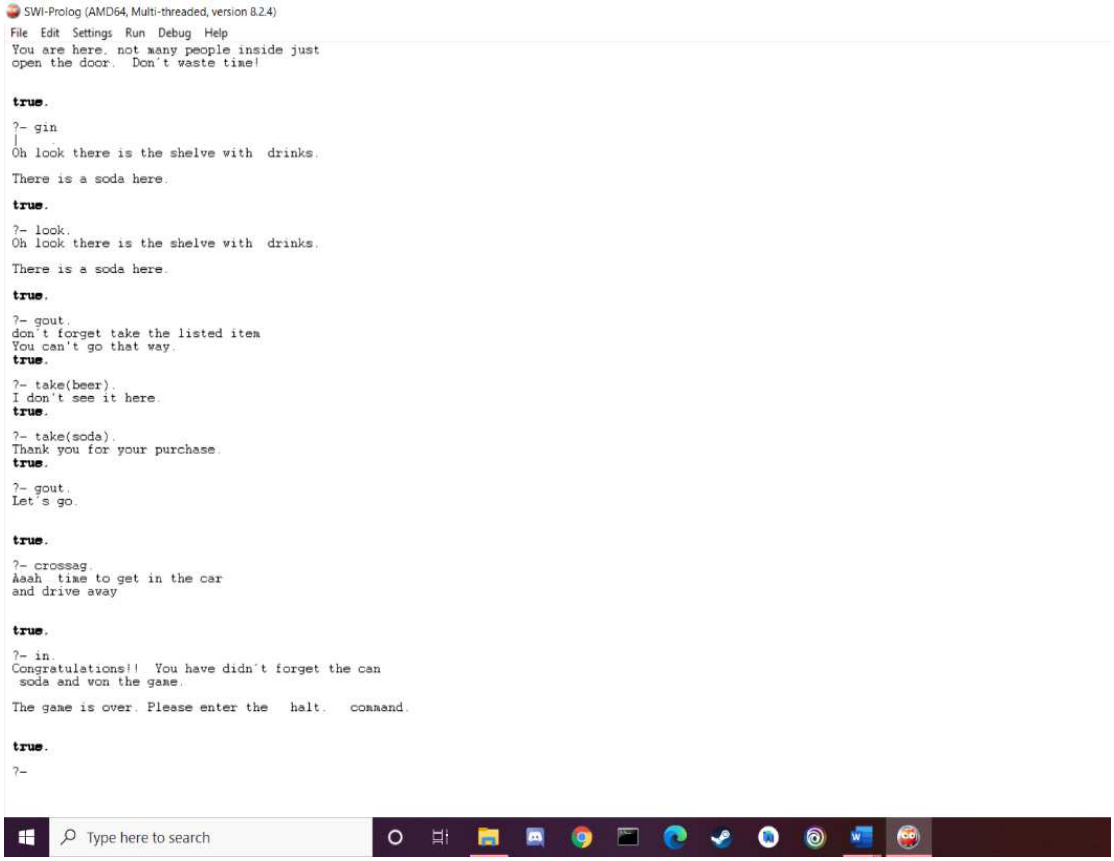
Τέλος, ορίζουμε τις διάφορες εκδοχές της describe, καθώς κάθε χώρος μπορεί να έχει παραπάνω από μία περιγραφές.

Παραδείγματα Εκτέλεσης:


SWI-Prolog (AMD64, Multi-threaded, version 8.2.4)
File Edit Settings Run Debug Help
Welcome to SWI-Prolog (threaded, 64 bits, version 8.2.4)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.
For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).
?-
% c:/Users/Andreas/Desktop/3191695-master/spider.pl compiled 0.00 sec, 34 clauses
?- start.
Enter commands using standard Prolog syntax.
Available commands are:
start. -- to start the game.
out.cross.gin.gout.crossag.in. -- to go there.
take(Object). -- to pick up an object.
open. -- to open doors.
look. -- to look around you again.
instructions. -- to see this message again.
halt. -- to end the game and quit.
You are in a car. Cross the road
enter the gas station;
take a can of soda
rand come back in
this car.
true.
?- out.
You are on the road cross it.
And go towards the gas station
true.
?- cross.
You are here, not many people inside just
open the door. Don't waste time!
true.
?-
Type here to search


SWI-Prolog (AMD64, Multi-threaded, version 8.2.4)
File Edit Settings Run Debug Help
out.cross.gin.gout.crossag.in. -- to go there.
take(Object). -- to pick up an object.
open. -- to open doors.
look. -- to look around you again.
instructions. -- to see this message again.
halt. -- to end the game and quit.
You are in a car. Cross the road
enter the gas station,
take a can of soda
rand cowe back in
this car.
true.
?- out.
You are on the road cross it.
And go towards the gas station
true.
?- cross.
You are here, not many people inside just
open the door. Don't waste time!
true.
?- gin
|
Oh look there is the shelve with drinks.
There is a soda here.
true.
?- look.
Oh look there is the shelve with drinks.
There is a soda here.
true.
?-
Type here to search


SWI-Prolog (AMD64, Multi-threaded, version 8.2.4)
File Edit Settings Run Debug Help
out.cross.gin.gout.crossag.in. -- to go there.
take(Object). -- to pick up an object.
open. -- to open doors.
look. -- to look around you again.
instructions. -- to see this message again.
halt. -- to end the game and quit.
You are in a car. Cross the road
enter the gas station.
take a can of soda
rand cowe back in
this car.
true.
?- out.
You are on the road cross it.
And go towards the gas station
true.
?- cross.
You are here, not many people inside just
open the door. Don't waste time!
true.
?- gin
|
Oh look there is the shelve with drinks.
There is a soda here.
true.
?- look.
Oh look there is the shelve with drinks.
There is a soda here.
true.
?- gout.
don't forget take the listed item
You can't go that way.
true.
?- take(beer).
I don't see it here.
true.
?- take(soda).
Thank you for your purchase.
true.
?-
Type here to search


SWI-Prolog (AMD64, Multi-threaded, version 8.2.4)
File Edit Settings Run Debug Help
You are here. not many people inside just
open the door. Don't waste time!
true.
?- gin
|
Oh look there is the shelve with drinks.
There is a soda here.
true.
?- look.
Oh look there is the shelve with drinks.
There is a soda here.
true.
?- gout.
don't forget take the listed item
You can't go that way.
true.
?- take(beer).
I don't see it here.
true.
?- take(soda).
Thank you for your purchase.
true.
?- gout.
Let's go.
true.
?- crossag.
Aaah time to get in the car
and drive away
true.
?- in.
Congratulations!! You have didn't forget the can
soda and won the game.
The game is over. Please enter the halt. command.
true.
?-
Type here to search