The Average Program

Example: write a program that asks the user for a list of integers, reads this list of integers from the terminal and computes the average value of the list.

```
lcount([],0).
lcount([_|T],C) :- lcount(T,C1), C is 1 + C1.

ladd([],0).
ladd([X|T],S) :- ladd(T,S1), S is X + S1.

ave(S,C,A) :- A is S/C.

lave(L,A) :- ladd(L,S),lcount(L,C),ave(S,C,A).

interact :-
          write('Enter a list> '),
          read(L),
          lave(L,A),
          write('The average is '),
          write(A),
          nl,
          nl,
          interact.
```

Quagent Prolog API

High level quagent interface

Action:

```
q_walk(+Quagent,+Distance)/2
q_turn(+Quagent,+Angle)/2
q_pickup(+Quagent,+Item)/2
q_drop(+Quagent,+Item)/2
```

Perception:

```
q_radius(+Quagent,+Radius)/2
q_rays(+Quagent,+No_of_Rays)/2
q_cameraon(+Quagent)/1
q_cameraoff(+Quagent)/1
```

Proprioception:

```
q_where(+Quagent)/1
q_inventory(+Quagent)/1
q_wellbeing(+Quagent)/1
```

Events:

```
q_events(+Quagent,-[Events])/2
```

Note:
+ input argument
- output argument

Quagent Prolog API

Low level quagent interface

```
q_connect(+Host,-QuagentDesc)/2
q_connect(-QuagentDesc)/1
q_close(+QuagentDesc)/1
q_write(+QuagentDesc,+String)/2
q_read(+QuagentDesc,-[Events])/2
```

Quagent Prolog Program

```
:- consult('quagent.pro').

print_events([]).
print_events([E|T]) :- write(E),nl,print_events(T).

simpleq :-
        q_connect(Q),
        q_walk(Q,100),
        q_events(Q,WalkEvents),
        print_events(WalkEvents),
        q_where(Q),
        q_events(Q,WhereEvents),
        print_events(WhereEvents),
        q_close(Q).
```

Make sure the WalkEvents and WhereEvents variables are distinct, otherwise Prolog will try to prove that both event lists are identical!!

turnq

Write a Quagent Prolog program that continuously walks a bit and takes right turns. Dangerous, we are not checking whether the quagent actually got to where we wanted it to go.

```
 :- consult('quagent.pro').

turnq(Q) :-
          q_walk(Q,100),
          q_turn(Q,-90),
          q_walk(Q,100),
          q_turn(Q,-90),
          q_walk(Q,100),
          q_turn(Q,-90),
          q_walk(Q,100),
          q_turn(Q,-90),
          turnq(Q).

run :-
          q_connect(Q),
          catch(turnq(Q),q_died,true),
          q_close(Q).
```