(+Stimuli, -Response) is det.

Matching: the built-in predicates = and \=

\= which is for not equal to

The cut, in Prolog, is a goal, written as !, which always succeeds, but cannot be backtracked. It is best used to prevent unwanted backtracking, including the finding of extra solutions by Prolog and to avoid unnecessary computations. The cut should be used sparingly.

An atom, in Prolog, means a single data item. start with a lower case letter.

match(+Mask, +Atom, ?Replacements) is nondet

Pattern matching. This emulation should be complete. Can be optimized

using caching of the pattern-analysis or doing the analysis at compile-time.

?- match(Tie,beer,T).

Tie = T, T = [].

atom_codes:

convert bet atom and listof characters(integer denoting characters)The built-in Prolog predicate atom_codes can convert an atom into the list of the numeric codes used internally to represent the characters in the atom, or vice-versa.

?- atom_codes(pizza,List).List = [112, 105, 122, 122, 97].

?- atom_codes(Store,[98,101,101,114]).Store = beer.

lookup(trie,key,-value)=== true if the term key is in Trie and associated with value

point{x:1,y:2} tag{key:value}

stream means input output

What does _ mean in Prolog?

dontcare. underscore, don't-care variable The Prolog variable _ (underscore) is a "don't-care" variable, which will match anything (atom, number, structure, …). For example, the rule. bad(Dog) :- bites(Dog, _). says that something (Dog) is bad if Dog bites anything.

atom An atom, in Prolog, means a single data item. It may be of one of four types: a string atom, like 'This is a string' or. a symbol, like likes , john , and pizza , in likes(john, pizza). Atoms of this type must start with a lower case letter.

read_word_list(Ws) :-

read_line_to_codes(user_input, Code), %read next line of input from stream %library Unify content of the lines as a list of character codes with Line after the line has been read. A line is ended by a newline character or end-of-file. Unlike read_line_to_codes/3, this predicate removes a trailing newline character.read_line_to_codes(+Stream, -Line:codes)

atom_codes(A, Code), %convert bet atom and listof characters(integer denoting characters)

% atom_length(Atom, Length) succeeds if Length unifies with the number of characters of the name of Atom.

%atom_chars(Atom, Chars) succeeds if Chars is the list of one-char atoms whose names are the successive characters of the name of Atom.

% atom_codes(Atom, Codes) is similar to atom_chars/2 but deals with a

% list of character codes.

tokenize_atom(A, Ws). %break text in into words,number,puctuation marks

/////debug//

leash(-all),spy(lookup).

?- chatbot.

? i am happy

* Call: (12) lookup(1, _G4620, _G4621)

* Exit: (12) lookup(1, [(1, _G4616)|_G4613], _G4616)

* Call: (15) lookup(1, [(1, [happy])|_G4613], _G4645)

* Exit: (15) lookup(1, [(1, [happy])|_G4613], [happy])

how long have you been happy ?

[trace] ?- match(Pattern, Dict, RT).

Call: (10) match(_9604, _9606, _9608) ? creep

Exit: (10) match([], _9606, []) ? creep

Pattern = RT, RT = [].

```
[trace]  ?-  match(S, Dict, Input)
|    .
   Call: (10) match(_6474, _6476, _6478) ? creep
   Exit: (10) match([], _6476, []) ? creep
S = Input, Input = [].



[debug]  ?- chatbot.
 * Call: (11) read_word_list(_2384)
|: hii
 * Exit: (11) read_word_list([hii])
hello .
 * Call: (13) read_word_list(_2544)



[debug]  ?- chatbot
|    .
 * Call: (12) read_util:read_line_to_codes(user_input, _21926)
|: hii
 * Exit: (12) read_util:read_line_to_codes(user_input, [104, 105, 105])
hello .
 * Call: (14) read_util:read_line_to_codes(user_input, _22086)
```

Debug Code:

leash(-all),trace.

leash(-all),spy(lookup).

I use leash(-all),trace here

// after you degug you get y point

//lines which finding the match of input

```
 Exit: (16) pattern([hii, 1], [hello, '.'])
   Call: (16) match([hii, 1], _18824, [hii])
   Call: (17) match([1], _18868, [])
   Call: (18) lookup(1, _18912, _18914)  %lookup(trie,key,value)

   Exit: (18) lookup(1, [(1, _18910)|_18904], _18910)
   Call: (18) lists:append(_18910, _19012, [])  ///_18910, is value
  Exit: (18) lists:append([], [], [])
   Call: (18) match([], [(1, [])|_18904], []) // _18904 dict repeating after this

   Exit: (18) match([], [(1, [])|_18904], [])
   Exit: (17) match([1], [(1, [])|_18904], [])

   Exit: (16) match([hii, 1], [(1, [])|_18904], [hii])

   Call: (16) match([hello, '.'], [(1, [])|_18904], _19278)
   Call: (17) match(['.'], [(1, [])|_18904], _19268)
   Call: (18) match([], [(1, [])|_18904], _19318)
   Exit: (18) match([], [(1, [])|_18904], [])
   Exit: (17) match(['.'], [(1, [])|_18904], ['.'])
   Exit: (16) match([hello, '.'], [(1, [])|_18904], [hello, '.'])
   Call: (16) reply([hello, '.'])
```

Call: (17) write(hello)

hello