```
;;;================================
;;; -*- Mode: Lisp; Syntax: Common-Lisp; -*-
;;; Code from Paradigms of Artificial Intelligence Programming
;;; Copyright (c) 1991 Peter Norvig
;;;
;;;  ELISA --  Pattern-Directed Dialog


(defun variable-p (x)
  "Is x a variable (a symbol beginning with `?')?"
  (and (symbolp x) (equal (char (symbol-name x) 0) #\?)))

(defconstant fail nil "Indicates pat-match failure")

(defconstant no-bindings '((t . t))
  "Indicates pat-match success, with no variables.")

;;; ==============================

(defun get-binding (var bindings)
  "Find a (variable . value) pair in a binding list."
  (assoc var bindings))

(defun binding-val (binding)
  "Get the value part of a single binding."
  (cdr binding))

(defun lookup (var bindings)
  "Get the value part (for var) from a binding list."
  (binding-val (get-binding var bindings)))

(defun extend-bindings (var val bindings)
  "Add a (var . value) pair to a binding list."
  (cons (cons var val)
        ;; Once we add a "real" binding,
        ;; we can get rid of the dummy no-bindings
        (if (and (eq bindings no-bindings))
            nil
            bindings)))

;;; ==============================

(defun pat-match (pattern input &optional (bindings no-bindings))
  "Match pattern against input in the context of the bindings"
  (cond ((eq bindings fail) fail)
        ((variable-p pattern)
         (match-variable pattern input bindings))
        ((eql pattern input) bindings)
        ((segment-pattern-p pattern)                ; ***
         (segment-match pattern input bindings))    ; ***
        ((and (consp pattern) (consp input))
         (pat-match (rest pattern) (rest input)
                    (pat-match (first pattern) (first input)
```

```
                                bindings)))
        (t fail)))

(defun match-variable (var input bindings)
  "Does VAR match input?  Uses (or updates) and returns bindings."
  (let ((binding (get-binding var bindings)))
    (cond ((not binding) (extend-bindings var input bindings))
          ((equal input (binding-val binding)) bindings)
          (t fail))))

(defun segment-pattern-p (pattern)
  "Is this a segment matching pattern: ((?* var) . pat)"
  (and (consp pattern)
       (starts-with (first pattern) '?*)))

;;; ==============================

(defun segment-match (pattern input bindings &optional (start 0))
  "Match the segment pattern ((?* var) . pat) against input."
  (let ((var (second (first pattern)))
        (pat (rest pattern)))
    (if (null pat)
        (match-variable var input bindings)
        ;; We assume that pat starts with a constant
        ;; In other words, a pattern can't have 2 consecutive vars
        (let ((pos (position (first pat) input
                             :start start :test #'equal)))
          (if (null pos)
              fail
              (let ((b2 (pat-match
                         pat (subseq input pos)
                         (match-variable var (subseq input 0 pos)
                                         bindings))))
                ;; If this match failed, try another longer one
                (if (eq b2 fail)
                    (segment-match pattern input bindings (+ pos 1))
                    b2)))))))

;;; ==============================

(defun rule-pattern (rule) (first rule))
(defun rule-responses (rule) (rest rule))

;;; ==============================

(defparameter *eliza-rules*
 '((((?* ?x) hello (?* ?y))
    (How do you do.  Please state your problem.))
   (((?* ?x) computer (?* ?y))
    (Do computers worry you?) (What do you think about machines?)
    (Why do you mention computers?)
    (What do you think machines have to do with your problem?))
   (((?* ?x) name (?* ?y))
    (I am not interested in names))
```

```
(((?* ?x) sorry (?* ?y))
 (Please don't apologize) (Apologies are not necessary)
 (What feelings do you have when you apologize))
(((?* ?x) I remember (?* ?y))
 (Do you often think of ?y)
 (Does thinking of ?y bring anything else to mind?)
 (What else do you remember) (Why do you recall ?y right now?)
 (What in the present situation reminds you of ?y)
 (What is the connection between me and ?y))
(((?* ?x) do you remember (?* ?y))
 (Did you think I would forget ?y ?)
 (Why do you think I should recall ?y now)
 (What about ?y) (You mentioned ?y))
(((?* ?x) if (?* ?y))
 (Do you really think its likely that ?y) (Do you wish that ?y)
 (What do you think about ?y) (Really-- if ?y))

(((?* ?x) I dreamt (?* ?y))
 (Really-- ?y) (Have you ever fantasized ?y while you were awake?)
 (Have you dreamt ?y before?))
(((?* ?x) dream about (?* ?y))
 (How do you feel about ?y in reality?))
(((?* ?x) dream (?* ?y))
 (What does this dream suggest to you?) (Do you dream often?)
 (What persons appear in your dreams?)
 (Don't you believe that dream has to do with your problem?))
(((?* ?x) my mother (?* ?y))
 (Who else in your family ?y) (Tell me more about your family))
(((?* ?x) my father (?* ?y))
 (Your father) (Does he influence you strongly?)
 (What else comes to mind when you think of your father?))

(((?* ?x) I want (?* ?y))
 (What would it mean if you got ?y)
 (Why do you want ?y) (Suppose you got ?y soon))
(((?* ?x) I am glad (?* ?y))
 (How have I helped you to be ?y) (What makes you happy just now)
 (Can you explain why you are suddenly ?y))
(((?* ?x) I am sad (?* ?y))
 (I am sorry to hear you are depressed)
 (I'm sure its not pleasant to be sad))
(((?* ?x) are like (?* ?y))
 (What resemblance do you see between ?x and ?y))
(((?* ?x) is like (?* ?y))
 (In what way is it that ?x is like ?y)
 (What resemblance do you see?)
 (Could there really be some connection?) (How?))
(((?* ?x) alike (?* ?y))
 (In what way?) (What similarities are there?))
(((?* ?x) same (?* ?y))
 (What other connections do you see?))

(((?* ?x) I was (?* ?y))
 (Were you really?) (Perhaps I already knew you were ?y)
```

```
 (Why do you tell me you were ?y now?))
(((?* ?x) was I (?* ?y))
 (What if you were ?y ?) (Do you thin you were ?y)
 (What would it mean if you were ?y))
(((?* ?x) I am (?* ?y))
 (In what way are you ?y) (Do you want to be ?y ?))
(((?* ?x) am I (?* ?y))
 (Do you believe you are ?y) (Would you want to be ?y)
 (You wish I would tell you you are ?y)
 (What would it mean if you were ?y))
(((?* ?x) am (?* ?y))
 (Why do you say "AM?") (I don't understand that))
(((?* ?x) are you (?* ?y))
 (Why are you interested in whether I am ?y or not?)
 (Would you prefer if I weren't ?y)
 (Perhaps I am ?y in your fantasies))
(((?* ?x) you are (?* ?y))
 (What makes you think I am ?y ?))

(((?* ?x) because (?* ?y))
 (Is that the real reason?) (What other reasons might there be?)
 (Does that reason seem to explain anything else?))
(((?* ?x) were you (?* ?y))
 (Perhaps I was ?y) (What do you think?) (What if I had been ?y))
(((?* ?x) I can't (?* ?y))
 (Maybe you could ?y now) (What if you could ?y ?))
(((?* ?x) I feel (?* ?y))
 (Do you often feel ?y ?))
(((?* ?x) I felt (?* ?y))
 (What other feelings do you have?))
(((?* ?x) I (?* ?y) you (?* ?z))
 (Perhaps in your fantasy we ?y each other))
(((?* ?x) why don't you (?* ?y))
 (Should you ?y yourself?)
 (Do you believe I don't ?y) (Perhaps I will ?y in good time))
(((?* ?x) yes (?* ?y))
 (You seem quite positive) (You are sure) (I understand))
(((?* ?x) no (?* ?y))
 (Why not?) (You are being a bit negative)
 (Are you saying "NO" just to be negative?))

(((?* ?x) someone (?* ?y))
 (Can you be more specific?))
(((?* ?x) everyone (?* ?y))
 (surely not everyone) (Can you think of anyone in particular?)
 (Who for example?) (You are thinking of a special person))
(((?* ?x) always (?* ?y))
 (Can you think of a specific example) (When?)
 (What incident are you thinking of?) (Really-- always))
(((?* ?x) what (?* ?y))
 (Why do you ask?) (Does that question interest you?)
 (What is it you really want to know?) (What do you think?)
 (What comes to your mind when you ask that?))
(((?* ?x) perhaps (?* ?y))
```

```
    (You do not seem quite certain))
   (((?* ?x) are (?* ?y))
    (Did you think they might not be ?y)
    (Possibly they are ?y))
   (((?* ?x))
    (Very interesting) (I am not sure I understand you fully)
    (What does that suggest to you?) (Please continue) (Go on)
    (Do you feel strongly about discussing such things?))))

;;; ==============================

(defun eliza ()
  "Respond to user input using pattern matching rules."
  (loop
    (print 'eliza>)
    (let* ((input (read-line-no-punct))
           (response (flatten (use-eliza-rules input))))
      (print-with-spaces response)
      (if (equal response '(good bye)) (RETURN)))))

(defun use-eliza-rules (input)
  "Find some rule with which to transform the input."
  (some #'(lambda (rule)
            (let ((result (pat-match (rule-pattern rule) input)))
              (if (not (eq result fail))
                  (sublis (switch-viewpoint result)
                          (random-elt (rule-responses rule))))))
        *eliza-rules*))

(defun switch-viewpoint (words)
  "Change I to you and vice versa, and so on."
  (sublis '((I . you) (you . I) (me . you) (am . are))
          words))

;;; ==============================

(defun read-line-no-punct ()
  "Read an input line, ignoring punctuation."
  (read-from-string
    (concatenate 'string "(" (substitute-if #\space #'punctuation-p
                                            (read-line))
                 ")")))

(defun punctuation-p (char) (find char ".,;:`!?#-()\\\""))

(defun print-with-spaces (list)
  (mapc #'(lambda (x) (prin1 x) (princ " ")) list))

(defun print-with-spaces (list)
  (format t "~{~a ~}" list))

;;; ==============================
```

```
(defun flatten (the-list)
  "Append together elements (or lists) in the list."
  (mappend #'mklist the-list))

(defun mklist (x)
  "Return x if it is a list, otherwise (x)."
  (if (listp x)
      x
      (list x)))

(defun mappend (fn the-list)
  "Apply fn to each element of list and append the results."
  (apply #'append (mapcar fn the-list)))

(defun random-elt (choices)
  "Choose an element from a list at random."
  (elt choices (random (length choices))))

(defun starts-with (list x)
  "Is x a list whose first element is x?"
  (and (consp list) (eql (first list) x)))

;;; ==============================

;;;? (eliza)
;;;ELIZA> hello
;;;HOW DO YOU DO. PLEASE STATE YOUR PROBLEM.
;;;ELIZA> my mother is hungry
;;;WHO ELSE IN YOUR FAMILY IS HUNGRY
;;;ELIZA> I am
;;;IN WHAT WAY ARE YOU
;;;ELIZA> very
;;;GO ON
;;;ELIZA> Gee i want something to eat
;;;WHY DO YOU WANT SOMETHING TO EAT
;;;ELIZA> I'm hungry
;;;DO YOU FEEL STRONGLY ABOUT DISCUSSING SUCH THINGS?
;;;ELIZA> yes, I want some candy
;;;WHAT WOULD IT MEAN IF YOU GOT SOME CANDY
;;;ELIZA> I'd be happy
;;;VERY INTERESTING
;;;ELIZA> quit
;;;GO ON
;;;ELIZA> exit
;;;I AM NOT SURE I UNDERSTAND YOU FULLY
;;;ELIZA> stop
;;;GO ON
;;;ELIZA> [Aborted]

;;; ==============================
```