Contest Problem – Pascal
Programming Contest, Spring 1991
Glassboro State College Department of Computer Science

Your program will implement what is called a "pattern matcher" – it will check an input word to see if it appears in a list of words, that is, if it matches. Here's an overall outline of the program:

1. Read in the word list, wordList. (from the keyboard or a file, as you wish.)
2. Echo wordList.
3. Read in a word to match (matchWord)
4. WHILE matchWord is not a sentinel to indicate the user is done, DO
5. FOR each word w in the wordList DO
6. If matchWord matches w THEN print out w
7. Print out the number of matches

The matching will allow the matchWord (but not the words in the wordList) to contain "Wild Cards". One ore more letters in the matchWord can be a wild card, which will be represented by '?'. The '?' matches *any* letter. An input word (matchWord) matches a word in the wordList (w) if all of the following are true:

1. The length of w and matchWord are the same.
2. For each letter in w, the corresponding letter in matchWord is either the same, or it's a wild card.

Examples:

Suppose wordList contains the following words: HOTDOG, HUNGER, HUNT, HUNTER.

If matchWord = HUNT:

HUNT does not match HOTDOG, since they have different lengths
HUNT does not match HUNGER, since they have different lengths
HUNT **does** match HUNT – all letters are the same
HUNT does not match HUNTER, since they have different lengths

If matchWord = HUN?ER:

HUN?ER does not match HOTDOG, since U is different than O
HUN?ER **does** match HUNGER, since HUN, ER match and ? matches the G
HUN?ER does not match HUNT – different lengths
HUN?ER **does** match HUNTER, since HUN, ER match and ? matches the T

If matchWord = H?????:

H????? **does** match HOTDOG, the H's match, and ????? matches OTDOG
H????? **does** match HUNGER, the H's match, and ????? matches UNGER
H????? does not match HUNT – different lengths
H????? **does** match HUNTER, the H's match, and ????? matches UNTER

The following data structures are recommended:

1. Each **word** is **array of char**; a sentinel can be used to mark the end of each word, if convenient.
2. The word list is **array of word** with each word as in 1. You can use a sentinel to mark the end of the word list, if convenient.

A good solution will result from a good top-down design. You might think about implementing procedures or functions for some of the following operations:

readWord(w:word) – reads a word from input
printWord(w:word) – prints one word
compareWords(w1: word; w2: word) – if w1 matches w2, return TRUE else FALSE
readWordList(list: wordList)

Assumptions:

1. All input is upper-case.
2. Each word is input on a line by itself, terminated by a period.
3. The word list is terminated by a line containing '#'.
4. No word is longer than 12 characters.
5. At most 100 words will ever be input to the program.

Sample data: Hand in a print out of a run of your program with the following data:

Wordlist:

BOGUS.
BONUS.
BURGER.
HOTDOG.
HUNGER.
HUNTER.
PASTA.
PENCIL.
PENNSYLVANIA.
PIE.
PIZZA.
#.

Words to search for:

BURGER
BURNER
BO?US
P???A
??????
?U??ER
P???