E-GENTING BUG HUNT 2015

General instructions:

1) Write testing procedures (i.e. a test plan) for the program described below.
2) Your testing procedures must be written in English language.
3) The competition is an open book test.
4) Electronic aids are not permitted.
5) The duration of the competition is 8 hours.
6) Do not discuss matters related to the competition with other contestants.
7) Your testing procedures may involve:
    a) manipulating the inputs of the program and verifying that the correct outputs are produced;
    b) inspection of the program to verify that it is programmed correctly;
    c) extracting functions or code fragments from the program and putting them in a program of your own creation that exercises the functions or code fragments in a way that would be difficult by manipulating the inputs of the complete program;
    d) any other process that verifies that the program is functioning correctly.
8) The words ‘must’, ‘must not’, ‘required’, ‘should’, ‘should not’, and ‘may’ are to be interpreted as described in RFC 2119 .

1 INTRODUCTION

Paradise Park is a theme park with roller coasters, water slides, parachute simulators and all the other paraphernalia of a archetypal theme park.

Paradise Park has been experiencing long queues of people at their information desk asking questions like 'where are the change rooms?' or 'where can I buy tickets for the roller coaster?' Paradise Park has been endeavouring to automate this process using a robot with speech recognition and speech synthesis.

The speech recognition infrastructure accepts a table of possible questions. When a question is spoken into the microphone, it returns the identity of the question that was most likely to have been asked or a code that indicates that the likelihood of any of the phrases being spoken was below the minimum required for reliability. When a question is recognised by the robot, it dictates the corresponding answer. When a question is not recognised, the robot makes a recording of the question so that the unrecognised questions can be manually reviewed and the database of questions and answers can be made more comprehensive.

However, maintaining the database of questions and answers is becoming burdensome because many of the questions are just slight variations on a theme and trigger a similar answer. For example the following questions should all trigger the answer 'Follow the blue line to the Big Dipper.'

> Where is the Big Dipper?
> Where is the roller coaster?
> Where is the coaster?
> How do I get to the Big Dipper?
> How do I get to the roller coaster?
> How do I get to the coaster?
> How do I find the Big Dipper?
> How do I find the roller coaster?
> How do I find the coaster?
> Etcetera.

What the programmers of the robot would like, is an input language processor that can accept a simple expression that describes all the variations in a compact and easily understood way and that automatically generates the table of questions and answers required by the robot. For example, they would like to be able to express the preceding alternatives and their common answer using an input rule similar to the following:

```
(Where is | How do I (get to | find)) the (Big Dipper | roller coaster | coaster):
        Follow the blue line to the Big Dipper;
```

The table used to program the robot is a simple text file consisting of a line of text for each question-answer pair with the question and answer separated by a tab character. For example:

```
Where is the Big Dipper <TAB> Follow the blue line to the Big Dipper
Where is the coaster <TAB> Follow the blue line to the Big Dipper
```

In addition to ordinary words, the speech recognition infrastructure interprets the special code ‘*’ to indicate ‘any text’. For example, the following question-answer pair is able to recognise questions with redundant expletives:

> Where is the * coaster <TAB> Follow the blue line to the Big Dipper.

The following sections define the explicit requirements of the input language processor.

2 INPUT GRAMMAR

The input language processor must initially convert the input file into a stream of tokens. The tokens are words (identified by ‘WORD’ in the BNF) and special characters. Words consist of a contiguous sequence of upper or lower case letters separated by special characters or white space The special characters are the colon (‘:’), semicolon (‘;’), vertical bar (‘|’), ampersand (‘&’), open bracket (‘(‘), close bracket (‘)’) and asterisk (‘*’). The white space characters are space, tab and new line. The input file may contain redundant white space characters at any point in the file except between the characters making up a word. The input language processor must ignore redundant white space characters.

Having converted the input file into a stream of tokens, the input language processor must verify that the token stream conforms to the grammar defined by the following yacc-format BNF:

```
input_file          : rule_list
                    ;

rule_list           : rule_list rule
                    | /* empty */
                    ;

rule                : question ':' answer ';'
                    ;

question            : alternative_list
                    ;

alternative_list    : alternative
                    | alternative_list '|' alternative
                    ;

alternative         : phrase
                    | key_phrase_set
                    ;

phrase              : component
                    | phrase component
                    ;
```

```
component           : WORD
                    | '*'
                    | '(' alternative_list ')'
                    ;

key_phrase_set      : phrase '&' phrase
                    | key_phrase_set '&' phrase
                    ;

answer              : word_list
                    ;

word_list           : WORD
                    | word_list WORD
                    ;
```

3 PROCESSING OF THE INPUT RULES

When the input language processor identifies a `rule` it must emit an output line for every sequence of words and asterisks (‘sequence’) that complies with the requirements of the `question` in the `rule`. Each output line must contain the sequence of words and asterisks that comply with the requirements of the `question` followed by a tab character followed by the words in the `answer`.

A sequence complies with the requirements of an `alternative_list` if the sequence complies with the requirements of any `alternative` in the `alternative_list`.

A sequence complies with the requirements of a `phrase` if the words and asterisks in the sequence are the same and in the same order as the `WORD` and `'*'` tokens in the `phrase` and the words and asterisks in any sequence that complies with the requirements of any `'(' alternative_list ')'` in the `phrase`.

A sequence complies with the requirements of a `key_phrase_set` if the sequence contains sub-sequences of words and asterisks that comply with the requirements of each phrase in the `key_phrase_set`. For example if the `key_phrase_set` was ‘where & roller & coaster’, then the sequences that would comply with the `key_phrase_set` would be:

    * where * roller * coaster *
    * where * coaster * roller *
    * roller * where * coaster *
    * roller * coaster * where *
    * coaster * where * roller *
    * coaster * roller * where *

4 OTHER REQUIREMENTS

The input language processor must reduce sequences of multiple asterisk characters in the output file to a single asterisk. For example, a simple expansion of ‘(how & get) (roller & coaster)’ might produce, among others, an output similar to ‘* how *get * * roller * coaster *’, but the input language process must reduce this to ‘*how * get * roller * coaster *’.

The input language processor should reduce sequences of multiple whitespace characters within an output line to a single space character.

When the input language processor encounters an invalid input sequence it should display a diagnostic message that indicates the input line number on which the error was detected and a description of the error. It should then skip tokens until it encounters a semicolon or the end of the input stream. If it encounters a semicolon, it should endeavour to restart the translation at the beginning of the next `rule`.

If a simple translation of an input file would emit multiple instances of the same question-tab-answer output line, the input language processor must reduce the multiple instances of the output line to a single instance.

If a simple translation of an input file would emit a single question with two or more different answers, the input language processor must display a diagnostic message that identifies the input line on which the duplicate answer was detected and must not output the line containing the duplicate answer.

When performing the comparison of questions and answers required to satisfy the preceding requirements, the input language processor must disregard the case of the characters in the questions and answers.