E-GENTING BUG HUNT 2016

General instructions:

1. Write testing procedures (i.e. a test plan) for the program described below.
2. Your testing procedures must be written in the English language.
3. Your testing procedures should have test cases which describes an input, action or event and the expected response.
4. The competition is an open book test.
5. Electronic aids are not permitted.
6. The duration of the competition is 8 hours.
7. Do not discuss matters related to the competition with other contestants.
8. 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.
9. The words ‘must’, ‘must not’, ‘required’, ‘should’, ‘should not’, and ‘may’ are to be interpreted as described in RFC 2119 .

1 INTRODUCTION

Standard Security provides a security service for high-rise office blocks and condominiums. Among other things, Standard maintains a computerised contact log in which they record information about each conversation they have with a tenant. For example, if a tenant reported a stranger wandering around corridors, Standard would make a note of this in the contact log.

The existing reporting functions only provide for the selection and listing of contact log entries by tenant name and the time of the log entry, however Standard has found searching through the contact log entries in this way very tedious and time consuming. They would now like a word-based search that lets them search through the contact log entries for particular words.

Your task is to draft testing procedures that determine whether the word-based search program created by a third party contractor, called 'Scour' conforms to the requirements of this specification.

2 USER INTERFACE

Scour must receive a list of search keys from its user. Typically, each search key is a word, but Scour must process search keys that contain punctuation characters. However, Scour should reject search keys containing white space characters. Scour may receive the list of words in the form of command line arguments.

Scour must produce a report with the following contents:

1. date and time the report was produced;
2. the search keys used to produce the report;
3. for each selected contact log entry:
    a. building name;
    b. unit number;
    c. tenant name;
    d. tenant's telephone number;
    e. guard's name;
    f. guard's telephone number;
    g. date and time of the contact log entry;
    h. contents of the contact log entry;
4. the number of selected contact log entries.

Scour should produce the report in HTML. Scour may emit the report contents to its standard output stream.

3 SELECTION

Scour must select all the contact log entries that contain one or more of the search keys. This is somewhat different from a typical Internet search engine, which typically selects the documents that contain all the search keys.

Scour must disregard case (e.g. ‘A’ should be treated the same as ‘a’) when determining whether a contact log entry contains a search key.

In addition there is a table of related words. The table of related words is a simple text file called ‘RELATED.TXT’. Each line in RELATED.TXT contains a space-separated list of related words. The related words have the same lexical rules as the search keys passed to Scour. If a user specifies a search key that is the same as one of the related words on a line in RELATED.TXT, Scour must select not only the contact log entries that contain the search key, but also the contact log entries that contain any of the other related words on the line in RELATED.TXT. Figure 1 contains an excerpt from RELATED.TXT.

```
dog dogs pooch canine canines
cat cats feline felines
car cars auto autos ride suv
```

Figure 1 - Excerpt from RELATED.TXT

The same word may appear on more than one line in RELATED.TXT. However, Scour should only search for the related words on lines containing the primary search key. It should not search for words related to words that are related to the primary search key.

4 DATABASE STRUCTURE

The database that contains the contact log entries contains the tables listed in Table 1.

Table 1 - Database Tables

Table Name	Description
`buildings`	Information related to each building that Standard Security protects.
`tenants`	Information related to each tenant of each building.
`guards`	Information related to each guard that Standard Security employs.
`logs`	Information related to each contact log entry.
`extensions`	Text extensions to the contact log entries.

The SQL schema entry for each table is listed below, followed by a description of each column.

```
create table buildings (
     buildingKey         integer not null,
     buildingName        char(80) not null
);
```

`buildingKey`
: is an integer that uniquely identifies the building.

`buildingName`
: is the name of the building.

```
create table tenants (
     tenantKey           integer not null,
     buildingKey         integer not null,
     unitNo              char(20) not null,
     tenantName          char(80) not null,
     tenantPhone         char(20) not null
);
```

`tenantKey`
: is an integer that uniquely identifies the tenant.

`buildingKey`
: is an integer, defined in the columns of the same name in `buildings`, that identifies the building.

`unitNo`
: is the number of the unit that the tenant occupies.

`tenantName`
: is the tenant's name.

`tenantPhone`
: is the tenant's telephone number.

```
create table guards (
     guardKey            integer not null,
     guardName           char(80) not null,
     guardPhone          char(20) not null
);
```

`guardKey`
: is an integer that uniquely identifies the guard.

`guardName`
: is the guard's name.

`guardPhone`
: is the guard's telephone number.

```
create table logs (
     logKey              integer not null,
     logTime             timestamp not null,
     tenantKey           integer not null,
     guardKey            integer not null,
     note                varchar(128)
);
```

`logKey`
: is an integer that uniquely identifies the contact log entry.

`logTime`
: is the date and time of the contact log entry.

`tenantKey`

is an integer, defined in the column of the same name in `tenants`, that identifies the tenant.

`guardKey`

is an integer, defined in the column of the same name in guards, that identifies the guard.

`note`

contains the first 128 characters of the textural contents of the contact log entry. If the contents of the contact log entry are less than or equal to 128 characters, then note contains the whole contact log entry. If the contact log entry is longer than 128 characters, `note` contains the first 128 characters and the remaining contents are stored in `extensions`.

```
create table extensions (
     logKey              integer not null,
     extNo               integer not null,
     extension           varchar(128) not null
);
```

`logKey`

is an integer, defined in the column of the same name in `logs`, that identifies the contact log entry to which the extension belongs.

`extNo`

is an extension number. If a contact log entry has more than one extension, each successive extension is assigned a progressively greater extension number.

`extension`

contains the next 128 characters (or less) of the textural contents of the contact log entry.

Each contact log entry has at least one row in logs, followed by zero or more rows in extensions. If the contact log entry contains, say 300 characters, the first 128 characters would be stored in `note` in `logs`, the second 128 characters would be stored in `extension` in `extensions` under, say, `extNo` 0, and the final 44 characters would be stored in `extension` in `extensions` under an `extNo` greater than 0, say 1.

The words in contact log entries may straddle the segment boundaries. For example, the first 3 characters in the word ‘confronted’ may be in `note` in `logs` and the remaining 7 characters may be in `extension` in `extensions`.