E-GENTING
PROGRAMMING COMPETITION 2015

General instructions:

1. Answer one or more of the questions.
2. The competition is an open book test.
3. The duration of the competition is 8 hours.
4. Do not discuss matters related to the questions with other contestants.
5. To receive credit for answering a question, your answer must be a credible response to the question. A credible response is an answer that solves the problem or would be likely to solve the problem with a little additional effort.
6. Provided your answer is a credible response, you will receive credit for the products of a methodical approach. For example, data flow diagrams and state transition diagrams and tables.
7. Your total score is the sum of the credit you receive from each credible response.
8. Your programs will be assessed on the ease with which they can be read and understood.
    - Indenting must be clean and consistent.
    - Variable names should describe the contents of the variables.
    - Coupling between modules should be visible.
    - Each module should do one thing well.
9. The questions are worth the following marks:

No	Name	Marks
1.	Database Restoration	250
2.	Approval Resolution	250
3.	Heart rate monitor	50
4.	Queue Display System	50

10. Unless otherwise stated, your programs may be written in any mainstream programming language under any mainstream operating system.
11. Unless otherwise stated, you may make use of all the standard library functions of your chosen language and operating system.
12. The words ‘must’, ‘must not’, ‘required’, ‘should’, ‘should not’, and ‘may’ are to be interpreted as described in RFC 2119[1].
13. You are NOT expected to answer all questions.

---

[1] *Key words for use in RFCs to Indicate Requirement Levels*, RFC 2119, S. Bradner, March 1997.

1 DATABASE RESTORATION

The Sultinex Shipping Company has been using an Anderbase DBMS for many years and has accumulated a large amount of historical information on their database. However, the corporation behind Anderbase was taken over several years ago and the new owners are no longer interested in supporting Anderbase-based systems. Further, several of Sultinex's tables contain close to the 2^{32} maximum number of rows in an Anderbase table.

Sultinex plan to convert their systems to a more up-to-date DBMS and would like a means for transferring the historical data on the Anderbase system to the new DBMS.

The Anderbase DBMS produces a binary backup file that Sultinex believe can be used to load the Anderbase data onto the new DBMS.

The Anderbase backup file contains tag-length-value envelopes that encapsulate fields. The structure of a tag-length-value envelope is represented in Figure 1. The first byte is a tag, which is a label that identifies the nature of the data that follows. The next two bytes contain the number of bytes in the value string. The two bytes are in big-endian format (i.e. $length_1$ contains the high-order bits and $length_2$ contains the low-order bits). This is followed by the bytes in the value string.

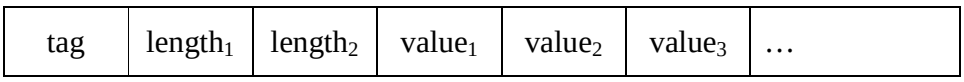


Figure 1 – Tag-Length-Value Envelope

For example, in the hexadecimal byte sequence '0A 00 04 12 34 56 78', the tag is hexadecimal byte 0A, the length is 00 04, or four bytes, and the value is four hexadecimal bytes 12, 34, 56 and 78.

The maximum number of bytes that may be stored in the value part of a tag-length-value envelope is 1023 (decimal) bytes. If a value is longer than 1023 bytes, the length field contains 1024 (decimal) and the value contains the first 1024 bytes to be stored. This initial tag-length-value envelope is then followed by one or more additional tag-length-value envelopes that contain the remaining data until the last envelope contains less than 1024 bytes. If the value contains an exact multiple of 1024 bytes, the length part of the last envelope is zero. A sequence of envelopes of this kind is called a 'chain'.

In the remaining part of this question, the nomenclature 'chain 0xNN' refers to a tag-length-value chain with tag number 0xNN, where 'NN' represents two hexadecimal digits.

The data dictionary in Figure 2 describes the format of the backup file.

If the column is an integer column, the column data contains either two or four binary bytes in two's complement big-endian format. If the column is a character column, the column data contains a string of ASCII characters.

The Sultinex database only contains integer and character data.

The character data only contains graphical ASCII characters (i.e. the characters with code point values from 32 to 126 decimal). The character data may contain single quote (') characters.

1. for each database table, a chain 0x01 containing:
    a. a chain 0x02 containing:
        i. a chain 0x03 containing the table name;
        ii. for each column in the table, a chain 0x04 containing:
            1. a chain 0x05 containing the column name;
            2. a chain 0x06 containing the data chain tag number;
            3. if the column is an integer column (SQL types INTEGER and SMALLINT), a chain 0x07 containing a single byte that contains the size of the integer in bytes (either 2 or 4);
            4. if the column is a character column (SQL type CHAR(n)), a chain 0x08 that contains two bytes in big-endian format that contain the capacity of the character string, 'n';
    b. a chain 0x09 containing:
        i. for each row in the table a chain 0x0a containing:
            1. for each column in the table a chain with the tag number defined in chain 0x06 above containing the data to be stored in the column (the 'column data').

Figure 2 – Backup File Data Dictionary

Your task is to write a program that reads the backup file created by the Anderbase DBMS, creates the tables in the Sultinex database and loads the tables with the data stored in the backup file.

Your program may use any mainstream DBMS interface. For example, ISO/IEC 9075 embedded SQL, JDBC or ODBC. Or alternatively, you can assume that a function 'SQL' is available that accepts a single string argument containing an SQL statement. For example, the following function call will create a table 'T' containing integer columns 'A' and 'B':

```
SQL ("create table T (A integer, B integer)");
```

To assist in understanding the structure of the backup file, the IT staff at Sultinex created a simple database using the SQL statements in Figure 3 and then listed the contents of the backup file using a hexadecimal dumping tool to produce the output in Figure 4.

```
create table orders (orderNo integer, orderName char(40))
insert into orders (orderNo, orderName)
     values (1, 'ORDER NAME 1')
insert into orders (orderNo, orderName)
     values (2, 'ORDER NAME 2')
```

Figure 3 – SQL Statements Used to Create Example Backup File

```
01 00 6e 02 00 36 03 00 06 6f 72 64 65 72 73 04  ..n..6...orders.
00 12 05 00 07 6f 72 64 65 72 4e 6f 06 00 01 10  .....orderNo....
07 00 01 04 04 00 15 05 00 09 6f 72 64 65 72 4e  ..........orderN
61 6d 65 06 00 01 11 08 00 02 00 28 09 00 32 0a  ame........(..2.
00 16 10 00 04 00 00 00 01 11 00 0c 4f 52 44 45  ............ORDE
52 20 4e 41 4d 45 20 31 0a 00 16 10 00 04 00 00  R NAME 1........
00 02 11 00 0c 4f 52 44 45 52 20 4e 41 4d 45 20  .....ORDER NAME
32                                               2
```

Figure 4 – Hexadecimal and Character Dump of the Example Backup file

2 APPROVAL RESOLUTION

The Jaystay Hotel Group manages a worldwide chain of four-star hotels. To facilitate the effective administration of the day-to-day business of the hotels, the board of directors has authorised executives and combinations of executives at various levels to enter into transactions on behalf of the group. These authorisations are defined in a table called the approval matrix. The approval matrix changes from time-to-time, but a typical approval matrix is presented in Table 1.

Table 1 – Typical Approval Matrix

Combination of Approvers	Maximum Purchase	Maximum Sale
EVP and two SVPs	10,000,000	20,000,000
EVP and one SVP	5,000,000	10,000,000
EVP	2,000,000	5,000,000
Two SVPs	2,000,000	5,000,000
SVP and two VPs	2,000,000	5,000,000
SVP and one VP	1,000,000	2,000,000
SVP	500,000	1,000,000
Two VPs	500,000	1,000,000
VP and two AVPs	500,000	1,000,000
VP and one AVP	250,000	500,000
VP	100,000	200,000
Two AVPs	100,000	200,000
AVP	not authorised	100,000

To assist secretarial-level staff in determining whether or not a transaction has been properly approved, Jaystay would like a program, the ‘approval resolution’ program into which they can type the names of the executives who have approved the transaction, whether the transaction is a purchase or a sale and the value of the transaction. Having received that information Jaystay would like the program to display ‘unapproved’, ‘partly approved’ or ‘fully approved’ depending on the status of the transaction.

Your task is to write the approval resolution program.

The names and levels of the potential approvers are stored in text file called EXECUTIVES.TXT, which contains one line for each executive. Each line contains the executive’s name followed by one or more tab characters followed by the executive’s level (either EVP, SVP, VP or AVP). The capitalisation of the executives’ names in

EXECUTIVES.TXT is not reliable. Some executive names are in lower case, some are in upper case and other names have the first letter capitalised and the others in lower case.

The approval matrix is stored in a text file called APPROVALS.TXT. The contents of APPROVALS.TXT conforms to the following yacc-format BNF:

```
approvals_file      : approvals_list
                    ;

approvals_list      : /* empty */
                    | approvals_list approval
                    ;

approval            : combination buy_opt sell_opt ';'
                    ;

combination         : quantity_opt level
                    | combination '+' quantity_opt level
                    ;

quantity_opt        : INTEGER '*'
                    | /* empty */
                    ;

level               : EVP | SVP | VP | AVP
                    ;

buy_opt             : BUY amount
                    | /* empty */
                    ;

sell_opt            : SELL amount
                    | /* empty */
                    ;

amount              : INTEGER
                    ;
```

The contents of APPROVALS.TXT corresponding to the approval matrix in Table 1 is presented in Figure 5.

APPROVALS.TXT may contain redundant white space characters at any point in the file except between the characters of a keyword. The white space characters are space, tab and new line.

The approval resolution program must interactively receive a list of approver names from the user. Although, technically, there is no limit to the number of executives who might approve a transaction, it is quite unlikely that a transaction would have more than 10 approvers.

The approval resolution program interactively receive a flag that indicates whether the transaction is a purchase or a sale and the amount of the transaction from the user.

When the approval resolution program has received the parameters mentioned in the preceding paragraphs, it must determine the status of the transaction and display the corresponding status phrase. The valid status phrases and their meanings are listed in Table 2.

The approval resolution program must disregard redundant approvals. For example if the transaction is a purchase of $250,000 and it is approved by two AVPs and a VP, the result is 'fully approved' despite the redundant AVP approval.

The approval resolution program must let the user enter the names of the approvers in any order.

The approval resolution program should reject approver names that are not in EXECUTIVES.TXT without requiring the user to re-enter any valid approver names that have already been entered and without requiring the user to restart the approval resolution program.

```
EVP + 2*SVP     BUY 10000000   SELL 20000000;
EVP + SVP       BUY 5000000    SELL 10000000;
EVP             BUY 2000000    SELL 5000000;
2*SVP           BUY 2000000    SELL 5000000;
SVP + 2*VP      BUY 2000000    SELL 5000000;
SVP + VP        BUY 1000000    SELL 2000000;
SVP             BUY 500000     SELL 1000000;
2*VP            BUY 500000     SELL 1000000;
VP + 2*AVP      BUY 500000     SELL 1000000;
VP + AVP        BUY 250000     SELL 500000;
VP              BUY 100000     SELL 200000;
2*AVP           BUY 100000     SELL 200000;
AVP                            SELL 100000;
```

Figure 5 – Example Contents of APPROVALS.TXT

Table 2 – Approval Resolution Program Status Phrases

Status Phrase	Meaning
unapproved	None of the approvers are empowered to approve the transaction.
partly approved	One or more of the approvers is empowered to approve the transaction in conjunction with another approver or approvers who have not yet approved the transaction.
fully approved	One or more combinations of the approvers is authorised to approve the transaction without any further approvals

3 HEART RATE MONITOR

Medical Devices Corporation (MDC) produces medical devices for hospitals. MDC released a new low-cost heart rate monitor around a month ago. Since the product launch, MDC has received multiple complaints about the new product. The primary complaint is about the inconsistency of the monitor. At this stage the problems have got so acute that customers are asking for refunds that are likely to cost millions of dollars. To prevent a big financial loss, MDC needs to fix the problem in the device. Unfortunately, the programmer for the device resigned just before the product launch. You are assigned to take over his job and troubleshoot the problem.

Part of the existing source code for the heart rate monitor is presented in Figure 6.

```
#include <stdio.h>
#include "heartio.h"
int
main (int argc, char *argv[])
{
     int rate;
     int ms;
     rate = 0;
     for (ms = 0; ms < 60*1000; ms += 10) {
          heart_sleep (10);
          if (heart_beat()) rate ++ ;
     }
     rate /= 60;
     printf ("%d\n", rate);
     for (;;) {
          heart_sleep(60*1000);
          rate += rand() % 3 – 1;
          printf ("%d\n", rate);
     }
}
```

Figure 6 – Faulty Source Code in Heart Rate Monitor

The program is defective in many ways. Firstly, it does not detect heartbeats correctly. Secondly, it randomly changed the heart rate instead of reading the actual heartbeat. An investigation into the Quality Control department is presently under way.

Your task is to rewrite the program to display the correct heart rate. The program must poll for the heart rate in the first minute and display the number of beats per minute on its standard output. For each second after that, the program must display the number of beats per minute gathered from the number of heartbeats in the preceding 60 seconds. The program should check the status of a heartbeat once every 10 milliseconds.

As a further complication, a new quality control team has discovered that the value returned by `heart_beat` does not turn on and off consistently. The function occasionally returns `false` in the middle of a heartbeat.

The root cause of the problem is a characteristic of the hardware, which uses a threshold blood pressure level to determine when a heartbeat is in progress. At the beginning and end of the heartbeat the blood pressure level is very close to the threshold and the output of `heart_beat` switches backwards and forwards between `true` and `false` before it switches consistently `true` or consistently `false`.

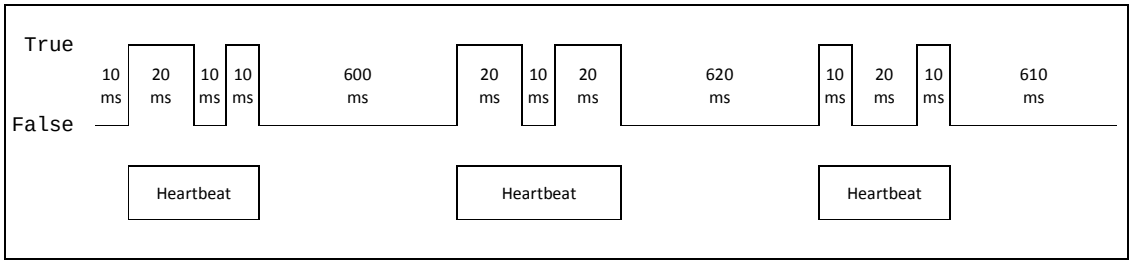


Figure 7 – Sample of Return Values that Indicate Heartbeats

The quality control team recommends that this problem be addressed using the classic switch de-bounce algorithm, the essence of which is that a heartbeat should be considered as having started on the first sample on which `heart_beat` returns `true`, but should not be considered as having ended until five 10ms samples all return `false`. The rewritten program should use the de-bounce algorithm to count heartbeats as recommended by the quality control team.

Since you are rewriting the whole program, you may use any mainstream programming language of your choice, though the program should use the API declared in Figure 8, Figure 9 or Figure 10 for timing and detecting heartbeats. If the language of your choice is not one of the three languages, you may adapt the API to your language of choice.

In the API, the `heart_sleep` function waits until a specific number of milliseconds have elapsed since last called or, if called for the first time, since the start of the program. If the duration has already elapsed when the function is called, it returns immediately. From the start of a heartbeat until the end of a heartbeat, the `heart_beat` function returns true. During any other time, it returns false.

```
// heartio.h

void heart_sleep(int ms);
                          // Sleep until the required number of
                          // milliseconds have elapsed since the last
                          // sleep or since the start of the program
bool heart_beat();
                          // Detect whether heartbeat is active and
                          // return true if a heartbeat is in progress
```

Figure 8 – C++ API Declarations

```
// Heartio.java

public class Heartio
{
      public static void heart_sleep(int ms);
                        // Sleep until the required number of
                        // milliseconds have elapsed since the last
                        // sleep or since the start of the program
      public static bool heart_beat();
                        // Detect whether heartbeat is active and
                        // return true if a heartbeat is in progress
}
```

Figure 9 – Java API declarations

```
// Heartio.cs

public class Heartio
{
      public static void heart_sleep(int ms);
                        // Sleep until the required number of
                        // milliseconds have elapsed since the last
                        // sleep or since the start of the program
      public static bool heart_beat();
                        // Detect whether heartbeat is active and
                        // return true if a heartbeat is in progress
}
```

Figure 10 – C# API Declarations

4 QUEUE DISPLAY SYSTEM

The Rupt Bank has a queue display system that is connected to 3 types of devices: a sequence number dispenser, a next customer pager and a paged customer display. The sequence number dispenser is a front-end device that allows customers to select a service and receive a sequence number for that service. Customer service representatives use the next customer pager to call the next customer to their counters. Each customer service counter has a next customer pager for this purpose. The paged customer display shows the latest sequence number being called and the calling counter.

When two counters call for customers at nearly the same time, only one of the calls is displayed. This caused many customers to miss their turn.

The bank wants to solve the problem by buying a new display that can show the latest 4 numbers. Your job is to write a new display driver program that displays the latest 4 numbers and their respective counters on the new display.

The new display is a 6x4 character display. It can show 4 rows of text and each row can contain 6 characters. The Rupt Bank would like the display to look something like the example in Figure 11.

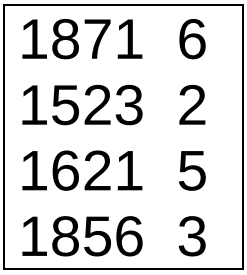


Figure 11 – Example Paged Customer Display

In this example, the most recent call is sequence number 1871 to counter 6.

The display hardware accepts the standard output of the display driver program and displays new output on the bottom most row, automatically scrolling the output upward and erasing the bottom line to make space for a new line when it receives a new line character. The top-most line is discarded when the display scrolls the text. The display can show the ten digits and the space character.

The display driver program must display the most recent call on the top of the display. Each display line must contain the four-digit sequence number, with leading digits if the number is less than 1000, followed by a space and then the single-digit counter number.

When the display driver program starts up, it must emit four new line characters to erase the display.

The display driver program must read the sequence number and counter number from its standard input, which is streamed from another program that interfaces with the devices at the customer service counters.

The input data stream consists of one line of data for each call. Each line of data contains 5 digits. The first 4 digits are the four-digit sequence number and the last digit is the single-digit counter number.

When the display driver program receives a valid input line containing a sequence number that is not already displayed, it must display the new sequence number and the corresponding counter number at the top of the display, scrolling the previous numbers down and dropping any number at the bottom of the display off the display.

If the display driver program receives a valid input line containing a sequence number that is already displayed, it must move the sequence number to the top row, show the counter number (which may have changed) and display the other numbers in their original order below the sequence number in the input line.

When the display driver program displays a new or updated sequence number as described in the preceding paragraphs, it must blink the sequence number, but not the counter number for 3 seconds from the time when the new or updated sequence number is shown. The blinking should consist of 700ms during which the number is shown and 300ms during which the number is not shown.

The display driver program may buffer new input in its standard input type-ahead buffer while it is flashing a new or updated sequence number.

If the display driver program receives an invalid input line (i.e. a line containing a non-numeric character or one that is too long or too short), it should quietly discard the invalid input line without changing the display.

PERTANDINGAN PENGATURCARAAN E-GENTING 2015

Arahan-arahan am:

1. Jawab satu atau lebih soalan yang diberikan.
2. Peserta-peserta dibenarkan mengguna buku rujukan.
3. Masa yang diperuntukan untuk pertandingan ini adalah 8 jam.
4. Perbincangan dengan peserta lain mengenai hal-hal soalan dan jawapan tidak dibenarkan.
5. Untuk menerima markah bagi jawapan untuk sesuatu soalan, jawapan anda mestilah merupakan penyelesaian yang munasabah bagi soalan tersebut. Penyelesaian yang munasabah ialah jawapan yang menyelesaikan masalah soalan atau jawapan yang mungkin menyelesaikan masalah soalan dengan sedikit usaha tambahan.
6. Sekiranya jawapan anda adalah penyelesaian yang munasabah, anda akan menerima markah untuk hasilan pendekatan teratur seperti gambar rajah aliran data, gambar rajah peralihan keadaan, jadual dan sebagainya.
7. Jumlah markah anda adalah jumlah markah yang anda perolehi dari setiap penyelesaian yang munasabah.
8. Program-program anda akan dinilai berdasarkan betapa mudahnya kod-kod sumber boleh dibaca dan difahami.
    - Takukan mestilah bersih dan sejajar.
    - Nama-nama pembolehubah harus menggambarkan isi kandungan pembolehubah-pembolehubah berkenaan.
    - Pergandingan di antara modul-modul mestilah nyata.
    - Setiap modul harus melakukan satu perkara dengan baik.
9. Markah yang diperuntukan kepada setiap soalan adalah seperti berikut:

No	Tajuk	Markah
1.	Pemulihan Pangkalan Data	250
2.	Resolusi Pelulusan	250
3.	Pemantau Denyutan Jantung	50
4.	Sistem Pemaparan Beratur	50

10. Kecuali dinyatakan, program anda boleh ditulis dengan menggunakan mana-mana bahasa pengaturcaraan utama di bawah mana-mana sistem operasi utama.
11. Kecuali dinyatakan, anda boleh menggunakan semua fungsi piawaian perpustakaan dalam bahasa pengaturcaraan dan sistem operasi yang anda pilih.
12. Perkataan-perkataan ‘must’, ‘must not’, ‘required’, ‘should’, ‘should not’, dan ‘may’ adalah ditafsirkan seperti diterangkan dalam RFC 2119[2].
13. Anda TIDAK dijangka untuk menjawab semua soalan.

[2] *Key words for use in RFCs to Indicate Requirement Levels*, RFC 2119, S. Bradner, March 1997.

1 PEMULIHAN PANGKALAN DATA

Syarikat Penghantaran Sultinex telah lama menggunakan DBMS Anderbase dan telah mengumpul maklumat bersejarah dalam jumlah yang besar di dalam pangkalan data mereka. Walau bagaimanapun, syarikat Anderbase telah diambil alih beberapa tahun yang lalu dan pemilik baru syarikat tersebut tidak berminat lagi dalam menyokong sistem-sistem yang berasaskan Anderbase. Tambahan pula, beberapa jadual Sultinex mengandungi bilangan rekod yang hampir mencecah bilangan rekod maksimum jadual Anderbase iaitu 2^{32} .

Sultinex berancang untuk menukar sistem-sistem mereka ke DBMS yang lebih baru dan ingin memindah data bersejarah yang berada dalam sistem Anderbase ke DBMS yang baru tersebut.

DBMS Anderbase menghasilkan fail backup binari yang Sultinex harap boleh dipergunakan untuk memuatkan data Anderbase ke dalam DBMS yang baru.

Fail backup Anderbase tersebut mengandungi sampul-sampul tag-panjang-nilai yang merangkumi medan-medan data. Struktur sampul tag-panjang-nilai tersebut diwakili dalam Figure 1. Bait pertama merupakan satu tag, iaitu satu label yang mengenalpasti sifat data yang berikutan. Dua bait seterusnya mengandungi bilangan bait dalam nilai rentetan. Dua bait ini adalah dalam format big-endian (iaitu $panjang_1$ mengandungi bit-bit aras tinggi dan $panjang_2$ mengandungi bit-bit aras rendah). Ini diikuti dengan bait-bait dalam nilai rentetan.

tag	$panjang_1$	$panjang_2$	$nilai_1$	$nilai_2$	$nilai_3$	…

Gambarajah 12 – Sampul Tag-Panjang-Nilai

Contohnya, dalam urutan bait perenambelasan (hexadesimal) ‘0A 00 04 12 34 56 78’, tag ialah bait hexadesimal 0A, panjang ialah 00 04, atau pun empat bait, dan nilai ialah empat bait hexadesimal 12, 34, 56 dan 78.

Bilangan maksimum bait yang boleh disimpan dalam bahagian nilai untuk satu sampul tag-panjang-nilai ialah 1023 (dalam sistem angka perpuluhan, atau desimal) bait. Jika sesuatu nilai lebih panjang daripada 1023 bait, medan panjang akan mengandungi 1024 (desimal) dan nilai mengandungi bait 1024 yang pertama untuk disimpan. Sampul tag-panjang-nilai yang pertama ini kemudiannya akan diikuti oleh satu atau lebih sampul-sampul tag-panjang-nilai tambahan yang mengandungi baki data sehingga sampul terakhir yang kurang daripada 1024 bait. Jika nilai mengandungi bait yang merupakan gandaan tepat bagi 1024 bait, bahagian panjang dalam sampul terakhir akan bernilai sifar. Urutan sampul-sampul sebegini digelar sebagai satu ‘rantai’.

Dalam penghuraian seterusnya dalam soalan ini, tatanama ‘rantai 0xNN’ merujuk kepada satu rantai tag-panjang-nilai dengan nombor tag 0xNN, di mana ‘NN’ mewakili dua digit hexadesimal.

Kamus data dalam Figure 2 menjelaskan format fail backup tersebut.

Jika sesebuah lajur merupakan lajur integer, data lajur tersebut mengandungi sama ada dua atau empat bait binari dalam format big-endian dua-pelengkap. Jika sesebuah lajur merupakan lajur aksara, data lajur tersebut mengandungi rentetan aksara ASCII.

Pangkalan data Sultinex hanya mengandungi data integer and aksara sahaja.

Data aksara tersebut hanya mengandungi aksara-aksara ASCII grafikal (iaitu aksara-aksara dengan nilai kod dari 32 hingga 126 dalam sistem desimal). Data aksara tersebut mungkin mengandungi aksara petikan tunggal (').

2. bagi setiap jadual pangkalan data, satu rantai 0x01 mengandungi:
    a. satu rantai 0x02 mengandungi:
        i. satu rantai 0x03 mengandungi nama jadual;
        ii. bagi setiap lajur dalam jadual, satu rantai 0x04 mengandungi:
            1. satu rantai 0x05 mengandungi nama lajur;
            2. satu rantai 0x06 mengandungi nombor tag rantai data;
            3. jika lajur merupakan satu lajur integer (SQL types INTEGER and SMALLINT), satu rantai 0x07 mengandungi satu bait yang mengandungi saiz integer dalam bait (sama ada 2 atau 4) ;
            4. jika lajur merupakan satu lajur aksara (SQL type CHAR(n)), satu rantai 0x08 yang mengandungi dua bait dalam format big-endian yang mengandungi kapasiti rentetan aksara tersebut, ‘n’;
    b. satu rantai 0x09 mengandungi:
        i. bagi setiap baris dalam jadual satu rantai 0x0a mengandungi:
            1. bagi setiap lajur dalam jadual satu rantai dengan nombor tag ditakrif dalam rantai 0x06 di atas mengandungi data yang perlu disimpan dalam lajur (‘lajur data’ tersebut).

Gambarajah 13 – Kamus Data Fail Backup

Tugas anda ialah mengaturcara satu program yang membaca fail backup tersebut yang dihasilkan oleh DBMS Anderbase, mewujudkan jadual-jadual tersebut dalam pangkalan data Sultinex dan memuatkan jadual-jadual tersebut dengan data yang tersimpan dalam fail backup tersebut.

Program anda boleh menggunakan mana-mana antaramuka DBMS yang utama. Contohnya, ISO/IEC 9075 embedded SQL, JDBC atau ODBC. Sebagai alternatif, anda boleh menganggap bahawa satu fungsi ‘SQL’ wujud boleh menerima satu argumen rentetan yang mengandungi satu kenyataan SQL. Contohnya, panggilan fungsi berikut akan mewujudkan satu jadual ‘T’ yang mengandungi lajur-lajur integer ‘A’ dan ‘B’:

```
SQL ("create table T (A integer, B integer)");
```

Untuk membantu dalam memahami struktur fail backup tersebut, kakitangan IT Sultinex telah menambah satu pangkalan data ringkas dengan menggunakan kenyataan dalam Figure 3 dan kemudian menyenaraikan kandungan fail backup tersebut dengan menggunakan satu perisian yang memuat turun kandungan pangkalan data tersebut dengan menggunakan format hexadesimal yang menghasilkan output dalam Figure 4.

```
create table orders (orderNo integer, orderName char(40))
insert into orders (orderNo, orderName)
     values (1, 'ORDER NAME 1')
insert into orders (orderNo, orderName)
     values (2, 'ORDER NAME 2')
```

Gambarajah 14 – Kenyataan-Kenyataan SQL Yang Diguna Untuk Menghasilkan Contoh Fail Backup

```
01 00 6e 02 00 36 03 00 06 6f 72 64 65 72 73 04  ..n..6...orders.
00 12 05 00 07 6f 72 64 65 72 4e 6f 06 00 01 10  .....orderNo....
07 00 01 04 04 00 15 05 00 09 6f 72 64 65 72 4e  ..........orderN
61 6d 65 06 00 01 11 08 00 02 00 28 09 00 32 0a  ame........(..2.
00 16 10 00 04 00 00 00 01 11 00 0c 4f 52 44 45  ............ORDE
52 20 4e 41 4d 45 20 31 0a 00 16 10 00 04 00 00  R NAME 1........
00 02 11 00 0c 4f 52 44 45 52 20 4e 41 4d 45 20  .....ORDER NAME
32                                               2
```

Gambarajah 15 – Hexadesimal dan Aksara muat turun bagi Fail Backup Contohan

2 RESOLUSI PELULUSAN

Kumpulan Hotel Jaystay menguruskan rangkaian hotel-hotel empat-bintang di seluruh dunia. Untuk memudahkan pengurusan bagi aktiviti harian hotel-hotel tersebut, lembaga pengarah telah memberi kuasa kepada eksekutif and kombinasi eksekutif-eksekutif pada pelbagai peringkat untuk melakukan transaksi bagi pihak Kumpulan tersebut. Pemberian kuasa ini ditakrif dalam satu jadual yang dinamakan matriks kelulusan. Matriks kelulusan ini berubah dari masa ke semasa, tetapi matriks kelulusan biasanya adalah seperti dalam Table 1.

Jadual 3 – Matriks Kelulusan Biasa

Kombinasi Eksekutif Yang Melulus	Pembelian Maksimum	Jualan Maksimum
EVP and dua SVP	10,000,000	20,000,000
EVP and satu SVP	5,000,000	10,000,000
EVP	2,000,000	5,000,000
Dua SVP	2,000,000	5,000,000
SVP and dua VP	2,000,000	5,000,000
SVP and satu VP	1,000,000	2,000,000
SVP	500,000	1,000,000
Dua VP	500,000	1,000,000
VP and dua AVP	500,000	1,000,000
VP and satu AVP	250,000	500,000

VP	100,000	200,000
Dua AVP	100,000	200,000
AVP	tidak dibenarkan	100,000

Untuk membantu kakitangan berpangkat setiausaha dalam menentukan sama ada sesuatu transaksi telah diluluskan dengan sewajarnya, Jaystay memerlukan satu program yang dinamakan ‘resolusi pelulusan’, yang menerima nama eksekutif yang meluluskan transaksi, sama ada transaksi tersebut merupakan satu pembelian atau penjualan, dan nilai transaksi tersebut. Setelah menerima maklumat tersebut Jaystay memerlukan program tersebut memaparkan ‘unapproved’, ‘partly approved’ atau ‘fully approved’ bergantung kepada status transaksi tersebut.

Tugas anda ialah mengaturcara program resolusi pelulusan tersebut.

Nama dan peringkat jawatan pelulus disimpan dalam satu fail teks yang dinamakan EXECUTIVES.TXT, yang mengandungi satu baris untuk setiap eksekutif. Setiap baris mengandungi nama eksekutif diikuti dengan satu atau lebih aksara tab, diikuti dengan peringkat jawatan eksekutif tersebut (sama ada EVP, SVP, VP atau AVP). Nama-nama eksekutif dalam EXECUTIVES.TXT tidak mengikut peraturan huruf besar atau kecil secara tetap. Sesetengah nama-nama eksekutif adalah dalam huruf kecil, manakala sesetengah adalah dalam semua huruf besar, atau huruf pertama dalam huruf besar dan yang lain dalam huruf kecil.

Matriks Kelulusan disimpan dalam satu fail teks yang dinamakan APPROVALS.TXT. Kandungan APPROVALS.TXT mematuhi yacc-format BNF berikut:

```
approvals_file      : approvals_list
                    ;

approvals_list      : /* empty */
                    | approvals_list approval
                    ;

approval            : combination buy_opt sell_opt ';'
                    ;

combination         : quantity_opt level
                    | combination '+' quantity_opt level
                    ;

quantity_opt        : INTEGER '*'
                    | /* empty */
                    ;

level               : EVP | SVP | VP | AVP
                    ;
```

```
buy_opt             : BUY amount
                    | /* empty */
                    ;

sell_opt            : SELL amount
                    | /* empty */
                    ;

amount              : INTEGER
                    ;
```

Kandungan APPROVALS.TXT yang sepadan dengan matriks kelulusan dalam Table 1 dibentangkan dalam Figure 5.

APPROVALS.TXT mungkin mengandungi aksara-aksara ruang putih yang berulang dalam mana-mana bahagian fail kecuali di antara aksara-aksara untuk sesuatu kata kunci. Aksara-aksara ruang putih tersebut ialah ruang ('space'), tab dan baris baru.

Program resolusi pelulusan tersebut mesti menerima senarai nama pelulus secara interaktif daripada pengguna. Walaupun dari segi teknikal, bilangan eksekutif untuk meluluskan satu transaksi adalah tidak terhad, kemungkinan sesuatu transaksi memerlukan lebih daripada 10 pelulus adalah amat kecil.

Program resolusi pelulusan tersebut mesti menerima satu nilai yang menandakan sama ada transaksi tersebut ialah satu pembelian atau penjualan, dan juga amaun transaksi tersebut secara interaktif daripada pengguna.

Apabila program tersebut telah menerima parameter seperti yang dijelaskan sebelum perenggan ini, program tersebut mesti menentukan status transaksi tersebut dan memaparkan frasa status yang sepadan. Frasa-frasa status yang sah dan maksud frasa-frasa tersebut disenaraikan dalam Table 2.

Program tersebut mesti mengabaikan pelulusan yang berlebihan. Contohnya, jika transaksi tersebut ialah satu pembelian bernilai $250,000 dan ia diluluskan oleh dua AVP dan satu VP, keputusannya ialah ‘fully approved’ walaupun terdapat pelulusan AVP yang berlebihan.

Progam tersebut mesti membenarkan pengguna memasukkan nama-nama pelulus dalam mana-mana susunan.

Program tersebut harus menolak nama-nama pelulus yang tidak terkandung dalam EXECUTIVES.TXT dengan cara yang tidak memerlukan pengguna memasukkan semula nama-nama pelulus yang sah yang telah dimasukkan sebelum penolakan tersebut, dan juga tanpa memulakan semula program tersebut.

```
EVP + 2*SVP     BUY 10000000   SELL 20000000;
EVP + SVP       BUY 5000000    SELL 10000000;
EVP             BUY 2000000    SELL 5000000;
2*SVP           BUY 2000000    SELL 5000000;
SVP + 2*VP      BUY 2000000    SELL 5000000;
SVP + VP        BUY 1000000    SELL 2000000;
SVP             BUY 500000     SELL 1000000;
2*VP            BUY 500000     SELL 1000000;
VP + 2*AVP      BUY 500000     SELL 1000000;
VP + AVP        BUY 250000     SELL 500000;
VP              BUY 100000     SELL 200000;
2*AVP           BUY 100000     SELL 200000;
AVP                            SELL 100000;
```

Gambarajah 16 – Contoh Kandungan APPROVALS.TXT

Jadual 4 – Frasa-frasa Status untuk Program Resolusi Pelulusan

Frasa Status	Maksud
unapproved	Tiada pelulus yang diberi kuasa untuk meluluskan transaksi tersebut.
partly approved	Satu atau lebih pelulus diberi kuasa untuk meluluskan transaksi tersebut dengan pelulusan daripada satu atau lebih pelulus yang lain, tetapi pelulus yang lain belum lagi meluluskan transaksi tersebut.
fully approved	Satu atau lebih kombinasi pelulus diberi kuasa untuk meluluskan transaksi tersebut tanpa memerlukan pelulusan tambahan.

3 PEMANTAU DENYUTAN JANTUNG

Medical Devices Corporation (MDC) menghasilkan alat-alat perubatan untuk hospital. Pada bulan yang lalu, MDC telah mengeluarkan satu alat baru yang digunakan untuk memantau kadar denyutan jantung. Semenjak pelancaran produk tersebut, MDC menerima banyak aduan terhadap produk tersebut. Aduan utama berkenaan dengan pemantau tersebut ialah bacaan yang tidak sejajar. Pada ketika ini, masalah ini amat serius sehingga pelanggan-pelanggannya meminta bayaran balik yang akan menyebabkan kerugian yang bernilai jutaan. Untuk mengelakkan kerugian ini, MDC perlu membetulkan masalah yang berlaku pada alat ini.

Malangnya, pengaturcara alat tersebut sudah meletak jawatan sebelum pelancaran produk tersebut. Anda ditugaskan untuk mengambil alih kerjanya dan menyelesaikan masalah ini.

Sebahagian kod yang sedia ada bagi pemantau denyutan jantung ini adalah seperti dalam Figure 6.

```
#include <stdio.h>
#include "heartio.h"
int
main (int argc, char *argv[])
{
     int rate;
     int ms;
     rate = 0;
     for (ms = 0; ms < 60*1000; ms += 10) {
          heart_sleep (10);
          if (heart_beat()) rate ++ ;
     }
     rate /= 60;
     printf ("%d\n", rate);
     for (;;) {
          heart_sleep(60*1000);
          rate += rand() % 3 – 1;
          printf ("%d\n", rate);
     }
}
```

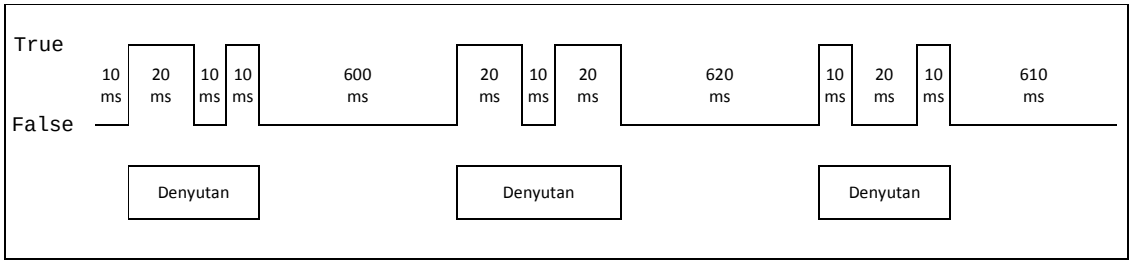
Gambarajah 17 – Kod Bermasalah dalam Pemantau Denyutan Jantung

Program ini cacat dari pelbagai segi. Yang pertamanya, ia tidak mengesan denyutan jantung dengan betul. Yang kedua, ia mengubah bacaan denyutan jantung secara rawak daripada bacaan yang sebenar. Satu siasatan telah dilancarkan ke atas jabatan Kawalan Kualiti.

Tugas anda ialah mengaturcara semula program tersebut supaya ia memaparkan bacaan denyutan jantung yang betul. Program tersebut mesti mengumpul bacaan denyutan jantung pada minit pertama dan memaparkan bilangan denyutan per minit pada output piawaian. Bagi setiap saat seterusnya, program tersebut mesti memaparkan bilangan denyutan per minit yang dikumpul dari bilangan denyutan dalam 60 saat sebelumnya. Program tersebut harus menyemak status denyutan jantung setiap 10 mili saat.

Kerumitan ditambah pula di mana satu pasukan kawalan kualiti baru telah mendapati bahawa nilai yang dipulangkan oleh `heart_beat` tidak berada dalam keadaan terpasang atau tertutup secara konsisten. Fungsi tersebut kadang-kala memulangkan `false` dalam pertengahan satu denyutan jantung.

Punca masalah tersebut ialah kerana ciri perkakasan tersebut yang menggunakan satu tahap tekanan darah ambang untuk menentukan sama ada sesuatu denyutan jantung sedang berlaku. Pada permulaan dan penamatan denyutan jantung, tahap tekanan darah adalah menghampiri tahap ambang dan kemudiannya output `heart_beat` berulang-alik di antara `true` dan `false` sebelum output tersebut menukar ke `true` atau `false` secara konsisten.



Gambarajah 18 – Contoh Nilai yang Menunjukkan Denyutan Jantung

Pasukan kawalan kualiti mengesyorkan supaya masalah ini ditangani dengan menggunakan algoritma 'classic switch de-bounce', di mana satu denyutan harus dianggap sebagai telah bermula pada sampel pertama apabila `heart_beat` memulangkan `true`, tetapi tidak harus dianggap sebagai telah tamat sehingga semua 5 sampel 10ms memulangkan `false.` Program baru tersebut harus menggunakan algoritma 'de-bounce' untuk menghitung denyutan seperti mana yang disyorkan oleh pasukan kawalan kualiti.

Memandangkan anda akan mengaturcara semula program tersebut, anda boleh menggunakan mana-mana bahasa pengaturacaraan utama pilihan anda, tetapi program tersebut harus menggunakan API yang diisytiharkan dalam Figure 8, Figure 9 atau Figure 10 untuk pengiraan masa dan pengesanan denyutan jantung. Jika bahasa pilihan anda bukan salah satu daripada tiga bahasa tersebut, anda boleh menyesuaikan API tersebut ke bahasa yang dipilih.

Dalam API tersebut, fungsi `heart_sleep` menunggu sehingga bilangan mili saat tertentu telah berlalu sejak panggilan terakhir, atau jika dipanggil pada kali pertama, sejak permulaan program tersebut. Jika tempoh tersebut telah tamat apabila fungsi itu dipanggil, fungsi tersebut akan pulang dengan serta-merta. Dari permulaan denyutan jantung sehingga penamatan denyutan jantung, fungsi `heart_beat` memulangkan `true`. Untuk masa yang lain, ia memulangkan `false`.

```
// heartio.h

void heart_sleep(int ms);
                          // Sleep until the required number of
                          // milliseconds have elapsed since the last
                          // sleep or since the start of the program
bool heart_beat();
                          // Detect whether heartbeat is active and
                          // return true if a heartbeat is in progress
```

Gambarajah 19 – Pengisytiharan API C++

```
// Heartio.java

public class Heartio
{
      public static void heart_sleep(int ms);
                        // Sleep until the required number of
                        // milliseconds have elapsed since the last
                        // sleep or since the start of the program
      public static bool heart_beat();
                        // Detect whether heartbeat is active and
                        // return true if a heartbeat is in progress
}
```

Gambarajah 20 – Pengisytiharan API Java

```
// Heartio.cs

public class Heartio
{
      public static void heart_sleep(int ms);
                        // Sleep until the required number of
                        // milliseconds have elapsed since the last
                        // sleep or since the start of the program
      public static bool heart_beat();
                        // Detect whether heartbeat is active and
                        // return true if a heartbeat is in progress
}
```

Gambarajah 21 – Pengisytiharan API C#

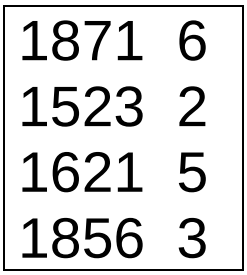
4 SISTEM PEMAPARAN BERATUR

Bank Rupt mempunyai satu sistem pemaparan beratur yang disambung kepada 3 jenis peranti: satu pengagih nombor urutan, satu pemanggil pelanggan berikutan dan satu paparan pelanggan yang dipanggil. Pengagih nombor urutan tersebut ialah satu peranti yang membenarkan pelanggan-pelanggan memilih satu perkhidmatan dan menerima satu nombor urutan bagi perkhidmatan tersebut. Wakil-wakil khidmat pelanggan menggunakan pemanggil pelanggan berikutan untuk memanggil pelanggan yang seterusnya ke kaunter mereka. Setiap kaunter khidmat pelanggan mempunyai satu pemanggil pelanggan berikutan untuk tujuan tersebut. Paparan pelanggan yang dipanggil memaparkan nombor urutan terkini yang dipanggil dan kaunter yang membuat panggilan tersebut.

Apabila dua kaunter memanggil pelanggan-pelanggan pada masa yang hampir sama, hanya salah satu panggilan tersebut dipaparkan. Ini menyebabkan banyak pelanggan terlepas giliran mereka.

Bank tersebut ingin menyelesaikan masalah ini dengan membeli satu paparan yang berupaya untuk memaparkan 4 nombor yang terkini. Tugas anda ialah mengaturcara satu program baru yang memaparkan 4 nombor terkini dan kaunter masing-masing pada paparan baru tersebut.

Paparan baru tersebut berupaya memaparkan 6x4 aksara. Ia berupaya untuk memaparkan 4 baris teks dan setiap baris mengandungi 6 aksara. Bank Rupt ingin paparan tersebut kelihatan seperti dalam Figure 11.



Gambarajah 22 – Contoh Paparan Pelanggan Yang Dipanggil

Dalam contoh ini, panggilan terakhir ialah nombor 1871 ke kaunter 6.

Perkakasan paparan tersebut menerima output piawaian bagi program tersebut dan memaparkan output baru pada baris yang paling bawah, menolak output semasa ke atas secara automatik dan memadam baris bawah supaya ada ruang untuk baris baru apabila ia menerima satu baris aksara baru. Baris paling atas akan dibuang apabila paparan menolak teks tersebut. Paparan tersebut berupaya untuk memaparkan sepuluh aksara digit dan aksara ruang.

Program tersebut mesti memaparkan panggilan terakhir pada baris paling atas dalam paparan tersebut. Setiap baris yang dipapar mesti mengandungi nombor urutan empat digit, dengan digit sifar di hadapan jika nombor tersebut adalah kurang daripada 1000, diikuti dengan satu ruang, dan kemudiannya nombor kaunter satu digit.

Apabila program tersebut dimulakan, ia mesti menghantar empat aksara baris baru supaya paparan tersebut dikosongkan.

Program tersebut mesti membaca nombor urutan dan nombor kaunter dari input piawaian yang diambil dari satu lagi program yang berantara-muka dengan peranti di kaunter-kaunter khidmat pelanggan.

Aliran data input tersebut mengandungi satu baris data bagi setiap panggilan. Setiap baris data mengandungi 5 digit. 4 digit pertama ialah nombor urutan dan digit terakhir ialah nombor kaunter.

Apabila program tersebut menerima satu baris input yang sah yang mengandungi nombor urutan yang belum dipaparkan, ia mesti memaparkan nombor urutan baru tersebut bersama dengan nombor kaunternya pada baris paling atas dalam paparan tersebut, dan menolak nombor sebelumnya ke bawah dan membuang nombor yang berada pada paling bawah daripada paparan tersebut.

Jika program tersebut menerima satu baris input sah yang mengandungi nombor urutan yang sudah dipaparkan, ia mesti memindah nombor urutan tersebut ke baris paling atas, memaparkan nombor kaunter (yang mungkin sudah berubah) dan memaparkan nombor-nombor lain dalam aturan yang sama di bawah nombor urutan baris input tersebut.

Apabila program tersebut memaparkan satu nombor urutan baru atau terkini seperti dijelaskan dalam perenggan sebelum ini, ia mesti mengelip nombor urutan tersebut, tetapi bukan nombor kaunter, selama 3 saat dari masa nombor urutan tersebut dipaparkan. Pengelipan tersebut harus menunjuk nombor tersebut selama 700ms dan menyembunyi nombor tersebut selama 300ms.

Program tersebut boleh menampan input baru dalam penampan ‘type-ahead’ input piawaiannya apabila ia sedang mengelip sesuatu nombor urutan baru atau terkini.

Jika program tersebut menerima satu baris input yang tidak sah (iaitu satu baris yang mengandungi aksara bukan nombor atau baris yang terlalu panjang atau terlalu pendek), ia harus mengabaikan baris input yang tidak sah tersebut tanpa mengubah kandungan dalam paparan.