E-GENTING
PROGRAMMING COMPETITION 2004

General instructions:

1. Answer one or more of the questions.
2. The competition is an open book test.
3. The duration for 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 will be the sum of the credit you received 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 marks allocated for the questions are as follows:

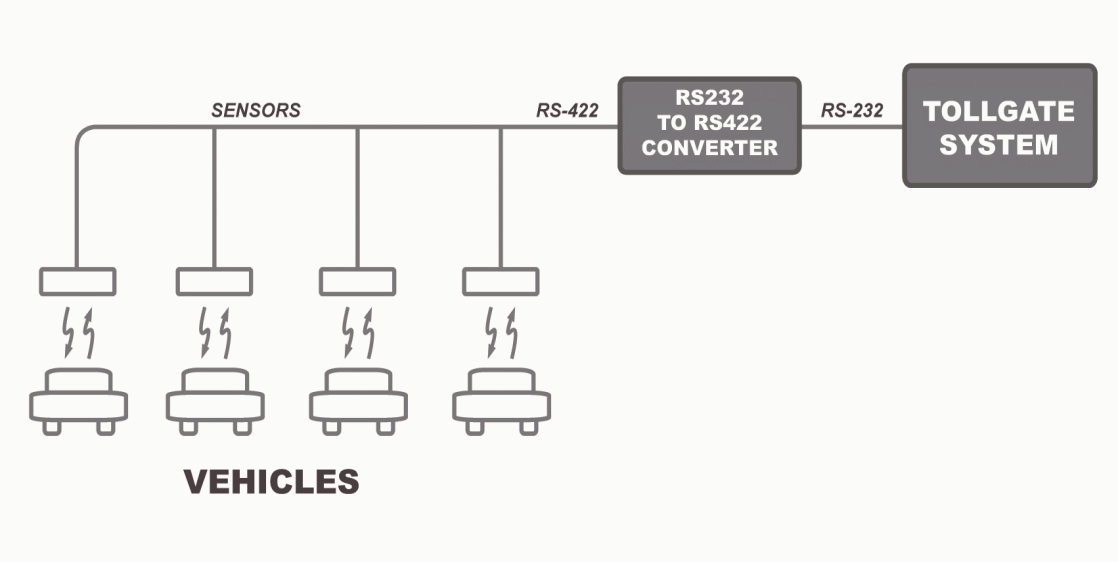
No	Name	Marks
1.	Tollgates	250
2.	Customer Profile	250
3.	Rounding	100
4.	Stock Price Display	400

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. You are NOT expected to answer all questions.

1 TOLLGATES

The operator of an expressway intends to trial some new sensors for toll collection. Each tollgate is equipped with a sensor. The sensors send a radio signal to transponders mounted on the inside of the windscreens of the passing vehicles. When polled, the transponders emit a response that is received by the sensor and passed back to a controlling computer.

The following diagram describes the layout of the proposed system.



The sensors are connected to the Tollgate System via an RS-422 serial network. RS-422 is a serial transmission standard with similar timing characteristics to RS-232, but different electrical characteristics. The electrical characteristics of RS-422 allow one RS-422 transmitter to drive up to 10 RS-422 receivers in parallel. RS-422 is easily converted to RS-232 and vice versa.

The sensors are polled by the Tollgate System, which must collect the vehicle identifiers of the passing vehicles and record them in a text file for later processing.

Your task is to program the Tollgate System in accordance with the following specifications.

The characteristics of the serial communications between the Tollgate System and the sensors are:

Speed	9600 baud
Data bits	8
Stop bits	1
Parity	none

The RS-422 network is wired up so that characters transmitted from the RS-232 port on the Tollgate System are received by all the sensors, but are not echoed back to the Tollgate System. The RS-232 port on the Tollgate System receives characters transmitted by any of the sensors. If two sensors attempt to transmit at the same time, the results are undefined.

All messages sent and received on the RS-422 network have the following general structure (where 'n' is the number of bytes in the message):

Byte Position	Contents
0	Destination address
1	Source address
2	Message type code
3	Body length
4 ... n-2	Body data
n-1	Longitudinal redundancy check (LRC)

A unique, 8-bit, non-zero, sensor address identifies each sensor. The sensor address is set by DIP-switches inside the sensor. The zero address identifies the Tollgate System.

The destination address is the address of device intended to receive the message. It may be either a sensor address or the zero Tollgate System address.

The source address is the address of the device that sent the message.

The message type code identifies the purpose of the message.

The body length is the length, in bytes, of the body data field.

The body data field is a variable-length field. Its contents are a function of the message type code.

The LRC is a checksum that results in a logical exclusive-or of all the bytes in the message from the destination address up to and including the LRC itself equalling zero.

The sensors are normally idle. To initiate a poll, the Tollgate System must send a Poll Command to one of the sensors. The format of the Poll Command is:

Byte Position	Data	Description
0	Sensor address	Destination address
1	0	Source address (i.e. Tollgate System address)
2	1	Message type code (POLL)
3	0	Body length
4	Computed	Longitudinal redundancy check

The sensors will respond to a Poll Command with either a Vehicle Detected Reply or a No-Vehicle Reply.

The Vehicle Detected Reply indicates that the transponder in a vehicle responded to the poll transmitted by the sensor. The format of the Vehicle Detected Reply is:

Byte Position	Data	Description
0	0	Destination address (i.e. Tollgate System address)
1	Sensor address	Source address

2	2	Message type code (VEHICLE_DETECTED)
3	16	Body length
4 ... 19	Vehicle id	Vehicle identifier
20	Computed	Longitudinal redundancy check

The No-Vehicle Reply indicates that the sensor did not receive a response to its poll. The format of the No-Vehicle Reply is:

Byte Position	**Data**	**Description**
0	0	Destination address (i.e. Tollgate System address)
1	Sensor address	Source address
2	3	Message type code (NO_VEHICLE)
3	0	Body length
4	Computed	Longitudinal redundancy check

The Tollgate System must store a list of sensor addresses. The sensor addresses should be stored in a static array in the source file.

The Tollgate System must poll each of the sensors one after the other by sending a Poll Command to each sensor in turn. It must wait for at least 200ms, but not more than 400ms, after completion of transmission of the last character in the Poll Command for a complete reply to be received. If a complete reply is not received in that time, the Tollgate System must time-out and go on to poll the next sensor in the sequence.

The Tollgate System should report any reply with an invalid LRC or other structural or message contents defect by logging a diagnostic message on the system console, but, in all other respects, it must ignore the defective reply.

In the event that a timeout, LRC fault or other structural or message contents defect is detected while receiving a reply, the Tollgate System must wait for at least 200ms, but not more than 400ms from the time the last character was sent to or received from the sensor network, before sending the next Poll Command. If a character is received during the wait, the wait must be restarted.

Similarly, when the Tollgate System is initiated, it must wait for a period of at least 200ms, but not more than 400ms, from the time of initiation of the system or the time of receipt of the last character from the sensor network, whichever occurs last, before it sends the first poll.

The Tollgate System must record the vehicle identifier of each vehicle that passes through the tollgates in a Transaction File. Each line of the Transaction File must contain the vehicle identifier of a vehicle that passed through the tollgate. Each byte in the vehicle identifier must be represented as two hex digits in the transaction file. For example, a listing of the Transaction File might look like this:

```
2c3720e51c1d71289205c03d76284809
b7d9377cdac3d8f3e72b86b66661575e
e7e1decb2ef7e2c859d2d72fd8c653ad
```

The sensors are polled at a sufficiently high rate that the transponder in a vehicle may reply to more than one poll. Consequently, the Tollgate System must identify repeated replies from the same vehicle and reduce them to a single entry in the Transaction File. If a repeated reply is received within 30 seconds from the last reply from the same vehicle, the repeated reply must not create a new Transaction File row. There must be an idle period of more than 30 seconds between any two replies from the same vehicle before the second of the two replies creates a new Transaction File row.

2 CUSTOMER PROFILE

Move-it-Quick, a freight forwarding and international courier business, uses a report known as the Customer Profile to determine the extent to which it is prepared to discount its services when a customer telephones in for a quote. The current implementation of the Customer Profile is fast enough for customers with a short transaction history, but can take a few minutes to generate for premium customers with significant numbers of transactions. Move-it-Quick's premium customers are getting tired of waiting and are hanging up while their Customer Profile is being generated. Move-it-Quick needs you to rewrite the program that generates the Customer Profile so that it can be generated in less than 5 seconds for any customer on the database.

The program that generates the Customer Profile accepts the customer identifier as a parameter and displays the following information:

1. date and time the report was generated;
2. customer identifier;
3. customer name;
4. customer's telephone number;
5. for each analysis period:
    a. the number of days in the analysis period;
    b. for each category of service that the customer purchased in the period:
        i. service category code,
        ii. number of sale transactions,
        iii. amount sold,
        iv. cost of providing the services,
        v. gross revenue;
    c. totals for all service categories of:
        i. number of sale transactions,
        ii. amount sold,
        iii. cost of providing the services,
        iv. gross revenue.

The analysis periods are:

1. the current day,
2. the past 30 days,
3. the past 182 days,
4. the past 365 days,
5. the past 730 days and
6. the total time over which the customer has traded with Move-it-Quick.

Gross revenue is the amount sold less the cost of providing the services.

The number of sale transactions in the totals may not be the same as the sum of the number of sale transactions for each service category because a single transaction may involve the provision of services from several categories.

The report must be formatted in accordance with the following layout:

```
         1         2         3         4         5         6         7         8
1...5....0....5....0....5....0....5....0....5....0....5....0....5....0....5....0
DD-MM-YY HH:MM                CUSTOMER PROFILE                          PAGE X

Customer id:       X------X
Customer name:     X--------------------------------------X
Telephone number:  X----------------------X

    Number  Service    Number        Amount       Cost of         Gross
   of days  category of trans          sold         sales       revenue
         1  X------X  XXX,XXX  X,XXX,XXX.XX  X,XXX,XXX.XX  X,XXX,XXX.XX
            X------X  XXX,XXX  X,XXX,XXX.XX  X,XXX,XXX.XX  X,XXX,XXX.XX
                      -------  ------------  ------------  ------------
            Total     XXX,XXX  X,XXX,XXX.XX  X,XXX,XXX.XX  X,XXX,XXX.XX

         7  X------X  XXX,XXX  X,XXX,XXX.XX  X,XXX,XXX.XX  X,XXX,XXX.XX
            X------X  XXX,XXX  X,XXX,XXX.XX  X,XXX,XXX.XX  X,XXX,XXX.XX
                      -------  ------------  ------------  ------------
            Total     XXX,XXX  X,XXX,XXX.XX  X,XXX,XXX.XX  X,XXX,XXX.XX

                      :          :             :             :

    XX,XXX  X------X  XXX,XXX  X,XXX,XXX.XX  X,XXX,XXX.XX  X,XXX,XXX.XX
            X------X  XXX,XXX  X,XXX,XXX.XX  X,XXX,XXX.XX  X,XXX,XXX.XX
                      -------  ------------  ------------  ------------
            Total     XXX,XXX  X,XXX,XXX.XX  X,XXX,XXX.XX  X,XXX,XXX.XX
```

The relevant parts of the database schema are:

```
// Customer File

create table custFile (
     custId         integer not null,
     custName       char(40) not null,
     custTel        char(25) not null
);
create unique index custIdInd on custFile(custId);

// Service File

create table servFile (
     servCode       char(16) not null,
     servCat        char(8) not null
);
create unique index servCodeInd on servFile(servCode);
```

```
// Transaction File

create table tranFile (
     tranNo         integer not null,
     tranCustId     integer not null,
     tranDate       integer not null
);
create unique index tranNoInd on tranFile(tranNo);
create index tranCustDateInd on
     tranFile(tranCustId, tranDate);

// Item File

create table itemFile (
     itemTranNo     integer not null,
     itemServCode   char(16) not null,
     itemAmtSold    double precision not null,
     itemCost       double precision not null
);
create unique index itemTranServInd on
     itemFile (itemTranNo, itemServCode);
```

`custFile`
: is a table that contains one row for each customer. There are around 100,000 customers.

`custId`
: is an 8-digit customer identifier.

`custName`
: contains the customer's name.

`custTel`
: contains the customer's telephone number.

`servFile`
: is a table that contains one row for each service. Move-it-Quick provides around 1,000 different services.

`servCode`
: is a code that identifies the service.

`servCat`
: is the service category code. Multiple services may belong to the same service category. There are 12 distinct service categories.

`tranFile`
: is a table that contains one row for each distinct transaction. The database contains 10 million Transaction File rows.

`tranNo`
: is a transaction number that uniquely identifies the transaction

`tranCustId`
: is the customer identifier of the customer with whom the transaction was conducted.

`tranDate`
: is the date on which the transaction was conducted in days since 1 January 1970. I.e. `tranDate` 60 is 2 March 1970.

`itemFile`
: is a table that contains one row for each distinct service that was provided to the customer in the transaction identified by the `itemTranNo` column. The database contains around 30 million Item File rows.

`itemTranNo`
: is the transaction number of the transaction in which the service was provided.

`itemServCode`
: is the service code that identifies the service that was provided.

`itemAmtSold`
: is the amount that was sold in cents (i.e. a sale amount of $12.34 would be stored as the floating point number 1,234.0).

`itemCost`
: is the cost of providing the service in cents.

Move-it-Quick has done some performance analysis on their current database and has recorded the execution times of the following SQL statements.

SQL Statement	**Execution Time**

`select custName, custTel`
`into :name, :tel`
`from custFile`
`where custId = :custId;`
	1.5ms

`select servCat`
`into :servCat`
`from servFile`
`where servCode = :servCode;`
	1.2ms

`declare cursor tranNoCur for`
`select tranNo`
`from tranFile`
`where tranCustId = :custId and`
`tranDate between :fromDate and :toDate;`
`// and fetch all the rows`
	2.0ms plus 0.5ms per row

`declare cursor itemDataCur for`
`select  itemServCode, itemAmtSold, itemCost`
`from  itemFile`
`where  itemTranNo = :tranNo;`
`// and fetch all the rows`
	1.2ms plus 0.7ms per row

`declare cursor itemDataCur for`
`select  itemTranNo, itemServCode,`
`  itemAmtSold, itemCost`
`from  itemFile;`
`// and fetch all the rows`
	0.4ms per row

Although the Transaction File contains 10 million rows, the transactions are not evenly distributed among the customers. In round numbers, 10 percent of the customers are responsible for 90 percent of the sales. A typical premium customer has 900 transactions on the database, each transaction having around three items. The best customer has 1,200 transactions on the database. Transactions are held on the database for 5 years.

Likewise, each customer has a concentration of preferred services. It would be unusual for any particular customer to purchase more than 50 different services.

3. ROUNDING

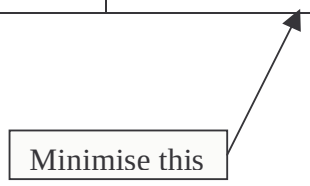
It is a common problem in commerce, that financial statistics stored to a precision of cents must be rounded to the nearest thousand dollars for presentation in company statements.

The difficulty is that if each unrounded statistic is rounded to the nearest thousand dollars, and the total is calculated by totalling the unrounded statistics and then rounding the total, the sum of the rounded statistics may not equal the rounded total. For example, if the individual statistics are four amounts of $100,250.00, the total rounded to the nearest thousand will be $401k, but the rounded individual statistics will each be $100k, which will add up to $400k, not $401k.

Your task is to write a method that accepts an array of unrounded statistics stored in units of cents in signed 32-bit integers and loads an array of the equivalent rounded statistics stored in units of thousands of dollars such that the total of the rounded statistics equals the rounded total of the unrounded statistics and the sum of the square of the rounding errors is minimised.

For example:

	Unrounded Statistics	Rounded Statistics	Rounding Errors
Indvidual statistics	4,400	5,000	600^2
	6.300	6,000	300^2
Total	10,700	11,000	450,000



4. STOCK PRICE DISPLAY

A firm of stockbrokers would like to set up a stock price display outside their office using a band of ultra-bright LEDs. The display is 16 LEDs high by 1024 LEDs long. It consists of 16 1024-bit shift registers loaded from the right-hand side of the display. Your task is to program a controller for the display.

The following function shifts the image on the display one pixel to the left, and loads the right most column of LEDs with new data:

In C and C++:

```
void ShiftAndLoad (unsigned short column);
```

In Java:

```
class DisplayControl {
    //...
    public static void shiftAndLoad (short column);
    //...
}
```

`column`
: is a bitmap that contains the state to be loaded into the right-most column of LEDs. If a bit is set, the corresponding LED will be illuminated. If a bit is reset, the corresponding LED will be switched off. Bits of increasing significance correspond to LEDs of increasing height. I.e. the least significant bit corresponds to the bottom LED and the most significant bit corresponds to the top LED.

The sequence to be displayed must be defined in a text file that is to be read by the control program during its initialisation.

Except as stated below, text in the text file must be shown on the display as-is.

New-line characters must be converted to space characters.

If the text file contains a sequence of the following form, the sequence must be converted into a stock price:

`<price server=`*`ip_address`*`:`*`port`* `code=`*`stock_code`*`>`

`ip_address`
: is the IP address of the stock price server in dot-format. For example: 192.0.1.5.

`port`
: is the port number of the stock price server. The '`:`*`port`*' part of the server address is optional. If it is omitted, the server port must default to 1845.

`stock_code`
: is a 15 character alphanumeric code that identifies the stock whose price is to be displayed. For example, the stock code might be GENTING for Genting Berhad.

For example, the sequence

```
<price server=192.0.1.5:1820 code=GENTING>
```
might be converted into the stock price 23.70.

Extra white space (i.e. space, tab and new-line characters) between the lexical elements of the sequence must be ignored.

The reserved words ‘price’, ‘server’ and ‘code’ must be accepted in upper case or lower case or a mixture of the two.

The text file may also contain the following special codes. The control program must recognise the codes and display the corresponding output:

Text File Code	Displayed Output
`&`	&
`<`	<

The control program must convert the characters to be displayed into bitmaps and then feed each column of the bitmap to the `ShiftAndLoad` function at a constant rate of 25 columns per second

When the display program gets to the end of the sequence, it must display a space character and then restart the sequence from the beginning.

The control program can obtain the bitmap corresponding to a particular character by calling the GetBitmap method described below.

In C and C++:

```
typedef struct {
     unsigned short           bmColCnt;
     const unsigned short     *bmColData;
} Bitmap_t;

const Bitmap_t *GetBitmap (char ch);
```

In Java:

```
class Bitmap {
     short               bmColCnt;
     short               bmColData[];
}
class DisplayControl {
     //...
     public static Bitmap getBitmap (char ch);
     //...
}
```

`bmColCnt`
: is the number of columns in the bitmap.

`bmColData`
: is a pointer (or reference) to an array of unsigned short variables, each containing the bitmap for one display column. Columns on the left side of the character have

low addresses in the array and columns on the right side of the character have high addresses in the array.

`ch`

is a character for which a bitmap is required.

`return value`

is a pointer (or reference) to a `Bitmap_t` structure that contains the parameters of the bitmap for the character identified by the `ch` parameter.

To obtain a stock price, the control program must establish a TCP connection to the server and then send a fixed-length request packet to the server. The format of the request packet is:

Byte Position	**Data**	**Description**
0 ... 1	1486	Request code (STOCK_PRICE_REQ)
2 ... 3	16	Body length
4 ... 19	Stock code	Stock code (null terminated)

The server will respond with the following response packet:

Byte Position	**Data**	**Description**
0 ... 1	1487	Response code (STOCK_PRICE_RESP)
2 ... 3	4	Body length
4 ... 7	Stock price	Stock price in cents

The byte order of all integers in the request and response packets is big endian.

When the control program is started, it must endeavour to obtain the stock price of each stock. It is not necessary for the control program to wait for all the stock prices to be obtained before it starts to display the running message. If a stock price has not been obtained by the time the control program needs to display it, the price may be shown as '?.??'.

After the initial poll for stock prices, the control program must attempt to update the price of each stock by sending another request to the stock's stock price server as soon as is practical after 60 seconds have elapsed since the previous stock price was obtained.

Some of the stock price servers return prices for more than one stock. The control program should endeavour to minimise the network overhead by sending the requests for all the stock prices obtained from a particular server one after another in a single TCP connection.

The physical connection to some of the stock price servers is unreliable. If a server fails to respond during the initial loading of the stock prices or if a stock price becomes out-of-date by more than five minutes, the stock price should be displayed as '?.??'.

If a connection cannot be made to a server, or if the connection to the server is broken during communications, the control program must abandon the connection and move on to update the next stock. It must then try and reconnect to the server as soon as is practical after 60 seconds have elapsed since the time of the failure.

PERTANDINGAN PENGATUCARAAN E-GENTING 2004

Arahan-arahan am:

1. Jawab satu atau lebih daripada soalan-soalan yang diberikan.
2. Pertandingan ini adalah satu ujian dimana calon-calon boleh merujuk kepada buku-buku atau bahan-bahan rujukan.
3. Masa yang diperuntukkan kepada pertandingan ini adalah lapan jam.
4. Perbincangan di antara peserta-peserta adalah tidak dibenarkan.
5. Kredit akan diberikan bagi hasil kerja dalam proses, khasnya, nota-nota seperti gambar rajah aliran data dan gambar rajah atau jadual alihan keadaan.
6. Kredit tambahan akan diberikan kepada mereka yang boleh menjawab lebih daripada satu soalan.
7. Jumlah markah anda adalah daripada jumlah kredit yang diperuntukkan kepada setiap soalan yang dijawab
8. Program-program anda akan dinilai berdasarkan kepada betapa mudahnya mereka boleh dibaca dan difahami.
    - Takukan mestilah bersih and sejajar.
    - Nama-nama pembolehubah harus menggambarkan isi kandungan pembolehubah-pembolehubah berkenaan.
    - Pasangan (*coupling*) di antara modul-modul mestilah nyata.
    - Setiap modul harus melakukan satu perkara dengan baik
9. Markah yang diperuntukkan kepada setiap soalan adalah seperti berikut:

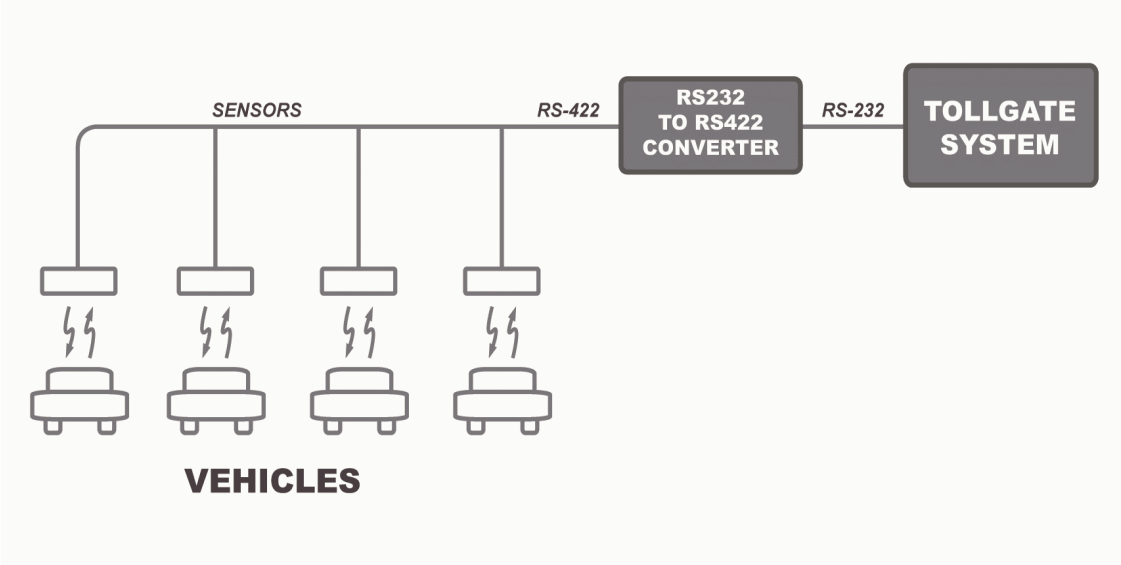
No	Nama	Markah
1.	Pintutol	250
2.	Profil Pelanggan	250
3.	Pembulatan	100
4.	Peranti Paparan Harga Stok	400

10. Anda harus menggunakan fungsi-fungsi perpustakaan piawai bagi bahasa pengaturcaraan yang dipilih.
11. Anda TIDAK dijangka menjawab semua soalan.

1. PINTUTOL

Sebuah syarikat yang menguruskan lebuh raya ingin mencuba beberapa penderia baru untuk pungutan tolnya. Setiap pintutol adalah diperlengkapi dengan satu penderia. Penderia-penderia itu menghantar satu isyarat radio ke alat isyarat yang diapasangkan di sebelah dalam cermin penahan angin kereta yang melaluinya. Apabila diberikan isyarat, alat isyarat akan memancarkan satu isyarat suap balik yang diterima oleh penderia dan dihantar seterusnya kepada komputer kawalan.

Gambar rajah dibawah menunjukkan susun atur sistem cadangan itu.



Penderia-penderia adalah dihubungkan kepada Sistem Pintutol menerusi satu rangkaian bersiri RS-422. RS-422 adalah satu piawai pancaran bersiri dengan sifat penentu masa yang seakan-akan sama dengan RS-232, tetapi dengan sifat elektrik yang berlainan. Sifat elektrik RS-422 membolehkan satu pemancar RS-422 menyokong sebanyak 10 penerima RS-422 secara selari. RS-422 boleh diubahsuaikan kepada RS-232 dan sebaliknya.

Penderia-penderia itu diberi isyarat oleh Sistem Pintutol. Mereka mesti mengumpul identiti kenderaan yang melalui mereka dan merekodnya dalam satu fail teks untuk diproses kemudian.

Tugas anda ialah mengatucarakan Sistem Pintutol mengikut spesifikasi yang berikut.

Ciri-ciri komunikasi bersiri diantara Sistem Pintutol dan penderia-penderia ialah:

Kelajuan	9600 baud
Bit data	8
Bit henti	1
Pariti	Tiada

Rangkaian RS-422 diwayarkan supaya aksara-aksara yang dipancarkan dari port RS-232 pada Sistem Pintutol diterima oleh semua penderia, tetapi tidak dipancarkan balik kepada Sistem Pintutol. Port RS-232 pada Sistem Pintutol merima aksara-aksara yang

dipancarkan oleh mana mana satu penderia. Jika dua penderia cuba memancar pada masa yang sama, keputusannya adalah tidak tertakrif.

Semua mesej yang dihantar dan diterima atas rangkaian RS-422 mempunyai struktur am yang berikut (dimana ‘n’ ialah bilangan bait dalam mesej):

Posisi Bait	**Kandungan**
0	Alamat destinasi
1	Alamat sumber
2	Kod jenis mesej
3	Kepanjangan bahagian utama
4 ... n-2	Data bahagian utama
n-1	Semakan lebihan membujur (SLM)

Satu alamat penderia unik, 8-bit, bukan-sifar menetukan setiap penderia. Alamat sesuatu penderia ditetapkan oleh suis DIP yang berada di dalam penderia itu. Alamat sifar merujuk kepada Sistem Pintutol.

Alamat destinasi ialah alamat alat yang dimaksudkan untuk menerima mesej. Ia boleh merujuk kepada alamat penderia atau alamat Sistem Pintutol yang bernilai sifar.

Alamat sumber ialah alamat alat yang menghantar mesej.

Kod jenis mesej menentukan tujuan mesej itu.

Kepanjangan bahagian utama ialah kepanjangan medan data bahagian utama dalam sebutan bait.

Medan data bahagian utama ialah satu medan panjang boleh ubah. Kandungannya ialah satu fungsi untuk kod jenis mesej itu.

SLM ialah satu hasil tambah semak yang diperolehi dengan melakukan operasi logik ATAU- eksklusif atas kesemua bait dalam mesej dari alamat destinasi hingga dan termasuk ia sendiri yang disamakan dengan sifar.

Biasanya, penderia-penderia itu adalah tidak berbuat apa-apa. Untuk memulakan satu perhubungan, Sistem Pintutol mesti menghantar satu Perintah *Poll* kepada salah satu daripada penderia-penderia itu. Format untuk Perintah *Poll* ialah:

Posisi Bait	**Data**	**Penghuraian**
0	Alamat penderia	Alamat destinasi
1	0	Alamat sumber (contohnya, alamat Sistem Pintutol)
2	1	Kod jenis mesej (POLL)
3	0	Kepanjangan tubuh
4	Terhitung	Semakan lebihan membujur (SLM)

Penderia-penderia itu akan membalas kepada satu Perintah *Poll* samada dengan satu Jawapan Kenderaan Dikesan atau Jawapan Tanpa Kenderaan.

Jawapan Kenderaan Dikesan menunjukkan bahawa alat isyarat yang terdapat di dalam satu kenderaan membalas kepada Perintah *Poll* yang dihantar kepadanya oleh penderia yang berkenanan. Format untuk Jawapan Kenderaan Dikesan ialah:

Posisi Bait	Data	Penghuraian
0	0	Alamay destinasi (iaitu alamat Sistem Pintutol)
1	Alamat penderia	Alamat sumber
2	2	Kod jenis mesej (VEHICLE_DETECTED)
3	16	Kepanjangan tubuh
4 ... 19	Pengecam kenderaan	Pengecam kenderaan
20	Terhitung	Semakan lebihan membujur (SLM)

Jawapan Tanpa Kenderaan menunjukkan bahawa penderia itu tidak menerima sebarang jawapan kepada Perintah *Poll*nya. Format untuk Jawapan Tanpa Kenderaan ialah:

Posisi Bait	Data	Penghuraian
0	0	Alamat destinasi (iaitu alamat Sistem Pintutol)
1	Alamat penderia	Alamat sumber
2	3	Kod jenis mesej (NO_VEHICLE)
3	0	Kepanjangan tubuh
4	Terhitung	Semakan lebihan membujur (SLM)

Sistem Pintutol mesti menyimpan satu senarai alamat penderia. Alamat penderia harus disimpan di dalam satu tatasusuan statik pada fail sumber.

Sistem Pintutol mesti menghubungi setiap penderia satu demi satu dengan menghantar satu Perintah *Poll* kepada setiap penderia secara bergilir-gilir. Ia mesti menunggu sekurang-kurangnya 200 milisaat, tetapi tidak melebihi 400 milisaat selepas menghantar aksara terakhir dalam Perintah *Poll*, untuk penerimaan jawapan yang sempurna. Jika jawapan lengkap tidak diterima dalam masa itu, Sistem Pintutol mesti *time-out* dan seterusnya menghubungi penderia yang berikut.

Sistem Pintutol itu mesti melaporkan apa-apa jawapan dengan LRC yang tidak sah atau kecacatan struktur atau kandungan mesej dengan mencatatkan satu mesej diagnostik pada konsol sistem, tetapi, dalam semua aspek yang lain, ia mesti mengabaikan jawapan cacat.

Sekiranya peristiwa *time-out*, kerosakan SLM, kecacatan struktur atau kecacatan kandungan mesej dikesan ketika sedang menerima satu jawapan, Sistem Pintutol mesti menunggu sekurang-kurangnya 200 milisaat, tetapi tidak melebihi 400 milisaat dari masa penghantaran aksara terakhir ke atau penerimaan aksara terakhir dari rangkaian penderia, sebelum menghantar Perintah *Poll* yang seterusnya. Jika satu aksara diterima sewaktu menunggu, penungguan itu mesti dimulakan sekali lagi.

Begitu juga, apabila Sistem Pintutol dimulakan, ia mesti menunggu sekurang-kurangnya 200 milisaat, tetapi tidak melebihi 400 milisaat, dari masa permulaan sistem itu atau masa penerimaan aksara yang terakhir dari rangkaian penderia, mana-mana yang berlaku terakhir, sebelum menghantar *poll* yang pertama.

Sistem Pintutol mesti merekod pengecam kenderaan untuk setiap kenderaan yang melalui pintutol-pintutol di dalam satu fail urus niaga. Setiap baris di dalam fail urus niaga mesti mengandungi pengecam kenderaan untuk satu kenderaan yang melalui pintutol. Setiap bait dalam pengecam kenderaan mesti diwakili dengan dua digit hex dalam fail urus niaga. Sebagai contoh, satu senarai fail urus niaga mungkin kelihatan seperti berikut:

```
2c3720e51c1d71289205c03d76284809
b7d9377cdac3d8f3e72b86b66661575e
e7e1decb2ef7e2c859d2d72fd8c653ad
```

Penderia-penderia adalah di*poll* pada kadar yang cukup tinggi sehingga alat isyarat di dalam satu kenderaan mungkin menjawab kepada lebih daripada satu *poll*. Akibatnya, Sistem Pintutol itu mesti mengesan jawapan-jawapan yang berulang kali daripada kenderaan yang sama dan mengurangkan mereka kepada satu masukan sahaja dalam Fail Urus Niaga. Jika satu jawapan yang berulang diterima dalam masa 30 saat selepas jawapan yang terakhir daripada kenderaan yang sama, jawapan itu tidak patut menghasilkan satu baris baru dalam Fail Urus Niaga. Keadaan tidak berbuat apa-apa lebih daripada 30 saat mestilah wujud diantara mana mana dua jawapan daripada kenderaan yang sama sebelum jawapan yang kedua menghasilkan satu baris baru dalam Fail Urus Niaga.

2. PROFIL PELANGGAN

Gerak-Pantas, satu syarikat antarabangsa pengangkut dan perniagaan penghantar, mengunakan satu laporan yang dikenali sebagai Profil Pelanggan untuk menentukan sebanyak mana ia bersedia memberi potongan harga bagi perkhidmatannya apabila pelanggan menalipon untuk sebut harga. Pelaksaanan Profil Pelanggan kini membolehkan penjanaan laporan bagi pelanggan-pelanggan yang mempunyai sejarah urus niaga yang singkat dengan cukup cepat, tetapi memakan masa beberapa minit untuk berbuat demikian bagi pelanggan-pelanggan utama yang mempunyai bilangan urus niaga yang bermakna. Pelanggan-pelanggan utama Gerak-Pantas merasa bosan untuk menunggu dan tersekat apabila Profil Pelanggan mereka sedang dijana. Gerak-Pantas memerlukan anda untuk menulis semula aturcara yang menjana Profil Pelanggan supaya ia boleh dilaksanakan dalam masa kurang daripada 5 saat bagi setiap pelanggan yang terdapat dalam pangkalan data.

Aturcara yang menghasilkan Profil Pelanggan menerima pengecam pelanggan sebagai satu parameter dan memaparkan butir-butir yang berikut:

1. tarikh dan masa laporan dijana;
2. pengecam pelanggan;
3. nama pelanggan;
4. nombor talipon pelanggan;
5. bagi setiap tempoh analisa:
    a. jumlah hari dalam tempoh tersebut;
    b. bagi setiap kategory pekhidmatan yang pelanggan terima dalam tempoh tersebut:
        i kod kategori perkhidmatan;
        ii bilangan urus niaga jualan;
        iii amaun jualan;
        iv kos bekalan perkhidmatan,
        v pendapatan kasar.
    c. jumlah berikut untuk semua kategori perkhidmatan:
        i bilangan urus niaga jualan;
        ii amaun jualan;
        iii kos bekalan perkhidmatan;
        iv pendapatan kasar.

Tempoh-tempoh analisa adalah:

1. hari semasa,
2. tempoh 30 hari yang lepas,
3. tempoh182 hari yang lepas,
4. tempoh 365 hari yang lepas,
5. tempoh 730 hari yang lepas dan
6. masa sepanjang mana pelanggan pernah berurus-niaga dengan Gerak-Pantas.

Hasil pendapatan kasar ialah amaun jualan tolak kos pembekalan perkhidmatan.

Jumlah bilangan urus niaga mungkin berbeza dengan hasil tambah bilangan urus niaga bagi setiap kategori perkhidmatan kerana satu urus niaga tunggal berkemungkinan melibatkan perkhidmatan daripada beberapa kategory.

Laporan itu mesti diformatkan menurut bentangan berikut:

```
         1         2         3         4         5         6         7         8
1...5....0....5....0....5....0....5....0....5....0....5....0....5....0....5....0
DD-MM-YY HH:MM                CUSTOMER PROFILE                          PAGE X

Customer id:       X------X
Customer name:     X-------------------------------------X
Telephone number:  X----------------------X

    Number  Service    Number        Amount       Cost of         Gross
   of days  category of trans          sold         sales       revenue
         1  X------X  XXX,XXX  X,XXX,XXX.XX  X,XXX,XXX.XX  X,XXX,XXX.XX
            X------X  XXX,XXX  X,XXX,XXX.XX  X,XXX,XXX.XX  X,XXX,XXX.XX
                      -------  ------------  ------------  ------------
            Total     XXX,XXX  X,XXX,XXX.XX  X,XXX,XXX.XX  X,XXX,XXX.XX

         7  X------X  XXX,XXX  X,XXX,XXX.XX  X,XXX,XXX.XX  X,XXX,XXX.XX
            X------X  XXX,XXX  X,XXX,XXX.XX  X,XXX,XXX.XX  X,XXX,XXX.XX
                      -------  ------------  ------------  ------------
            Total     XXX,XXX  X,XXX,XXX.XX  X,XXX,XXX.XX  X,XXX,XXX.XX

                      :          :             :             :

    XX,XXX  X------X  XXX,XXX  X,XXX,XXX.XX  X,XXX,XXX.XX  X,XXX,XXX.XX
            X------X  XXX,XXX  X,XXX,XXX.XX  X,XXX,XXX.XX  X,XXX,XXX.XX
                      -------  ------------  ------------  ------------
            Total     XXX,XXX  X,XXX,XXX.XX  X,XXX,XXX.XX  X,XXX,XXX.XX
```

Bahagian yang berkenaan dengan skema pangkalan data adalah:

```
// Customer File

create table custFile (
     custId         integer not null,
     custName       char(40) not null,
     custTel        char(25) not null
);
create unique index custIdInd on custFile(custId);

// Service File

create table servFile (
     servCode       char(16) not null,
     servCat        char(8) not null
);
create unique index servCodeInd on servFile(servCode);
```

```
// Transaction File

create table tranFile (
     tranNo         integer not null,
     tranCustId     integer not null,
     tranDate       integer not null
);
create unique index tranNoInd on tranFile(tranNo);
create index tranCustDateInd on
     tranFile(tranCustId, tranDate);

// Item File

create table itemFile (
     itemTranNo     integer not null,
     itemServCode   char(16) not null,
     itemAmtSold    double precision not null,
     itemCost       double precision not null
);
create unique index itemTranServInd on
     itemFile (itemTranNo, itemServCode);
```

`custFile`
: ialah satu jadual yang mengandungi satu baris untuk setiap pelanggan. Di sini, terdapatnya lebih kurang 100,000 pelanggan.

`custId`
: ialah satu pengecam pelanggan 8-digit.

`custName`
: mengandungi nama pelanggan.

`custTel`
: mengandungi nombor talipon pelanggan.

`servFile`
: ialah satu jadual yang mengandungi satu baris untuk setiap jenis perkhidmatan. Gerak-Pantas menawarkan lebih kurang 1,000 jenis perkhidmatan yang berlainan.

`servCode`
: ialah satu kod yang mengenalpasti jenis perkhidmatan.

`servCat`
: ialah kod kategori perkhidmatan. Pelbagai jenis perkhidmatan mungkin adalah kepunyaan kategori perkhidmatan yang sama. Di sini, terdapat sebanyak 12 kategori perkhidmatan yang berlainan.

`tranFile`
: ialah satu jadual yang mengandungi satu baris untuk setiap urus niaga berlainan. Pangkalan data mengandungi 10 juta baris dalan Fail Urus Niaga.

`tranNo`
: ialah satu nombor urus niaga yang mengenalpasti urus niaga.

`tranCustId`
: ialah pengecam pelanggan bagi pelanggan yang menjalankan urus niaga.

`tranDate`
: ialah tarikh dimana urus niaga dilaksaankan dalam sebutan berapa hari sejak 1 Januari 1970. Sebagai contoh, `tranDate` 60 ialah 2 Mac 1970.

`itemFile`
: ialah satu jadual yang mengandungi satu baris bagi setiap perkhidmatan berlainan yang diberikan kepada pelanggan dalam urus niaga yang dikenalpasti dengan `itemTranNo`. Pangkalan data mengandungi lebih kurang 30 juta baris dalam itemFile.

`itemTranNo`
: ialah nombor urus niaga bagi urus niaga dimana perkhidmatan diberikan.

`itemServCode`
: ialah kod perkhidmatan yang mengecam jenis perkhidmatan yang diberikan.

`itemAmtSold`
: ialah amaun jualan dalam sen (iaitu satu amaun jualan $12.34 akan disimpan sebagai nombor titik terapung 1,234.0).

`itemCost`
: ialah kos untuk pembekalan perkhidmatan dalam sen.

Gerak-Pantas telah membuat beberapa analisa pencapaian atas pangkalan data semasa dan merekodkan tempoh laksanaan bagi kenyataan-kenyataan SQL berikut:

Kenyataan SQL	**Tempoh Laksanaan**

`select custName, custTel`
`into :name, :tel`
`from custFile`
`where custId = :custId;`
	1.5 milisaat

`select servCat`
`into :servCat`
`from servFile`
`where servCode = :servCode;`
	1.2 milisaat

`declare cursor tranNoCur for`
`select tranNo`
`from tranFile`
`where tranCustId = :custId and`
`tranDate between :fromDate and :toDate;`
`// and fetch all the rows`
	2.0 milisaat campur 0.5 milisaat per baris

`declare cursor itemDataCur for`
`select  itemServCode, itemAmtSold, itemCost`
`from  itemFile`
`where  itemTranNo = :tranNo;`
`// and fetch all the rows`
	1.2 milisaat campur 0.7 milisaat per baris

`declare cursor itemDataCur for`
`select  itemTranNo, itemServCode,`
`  itemAmtSold, itemCost`
`from  itemFile;`
`// and fetch all the rows`
	0.4 milisaat per baris

Walaupun Fail Urus Niaga mengandungi 10 juta baris, rekod-rekod urus niaga tidaklah bertabur secara seimbang diantara pelanggan-pelanggan. Dalam sebutan genap, 10 peratus pelanggan adalah bertanggungjawap keatas 90 peratus jualan. Satu pelanggan utama biasanya mempunyai 900 urus niaga dalam pangkalan data, setiap urus niaga melibatkan lebih kurang tiga jenis perkhidmatan. Pelanggan yang paling baik mempunyai 1,200 urus niaga dalam pangkalan data. Urus niaga adalah disimpan dalam pangkalan data selama 5 tahun.

Seperti satu sama lain, setiap pelanggan menumpu kepada jenis-jenis perkhidmatan yang digemarinya sahaja. Adalah luarbiasa bagi mana-mana pelanggan membeli lebih daripada 50 jenis perkhidmatan yang berlainan.

.

3. PEMBULATAN (*ROUNDING*)

Ini adalah satu masalah lazim di bidang perdagangan dimana statistik-statistik kewangan yang disimpan sehingga kepersisan sen mestilah dibulatkan kepada ribu ringgit yang terdekat untuk persembahan di penyata kewangan syarikat.

Masalah yang dihadapi ialah jika setiap statistik bukan nombor bulat dibulatkan kepada ribu ringgit yang terdekat, dan jumlah adalah dikira dengan mencampurkan statistik-statistik asal diikuti dengan pembulatan, hasil tambah statistik-statistik yang telah dibulatkan mungkin berbeza daripada jumlah statistik-statistik asal yang telah dibulatkan. Contohnya, jika statistik-statistik individu adalah 4 amaun yang sama iaitu $100,250.00, jumlah numbor-nombor ini adalah $401 ribu setelah dibulatkan kepada ribu ringgit yang terdekat. Tetapi, setiap statistik individu akan menjadi $100 ribu selepas pembulatan dan hasil tambah mereka akan menjadi $400k, bukan $401k.

Tugas anda adalah menulis satu kaedah (*method*) yang menerima satu tatasusunan integer-integer bertanda 32-bit yang dimuatkan dengan statistik-statistiks yang disimpan dalam unit sen dan memuatkan satu tatasusunan lain dengan statistik-statistik setara yang disimpan dalam unit ribu ringgit dengan syarat-syarat yang berikut:

i Hasil tambah statistik-statistik setara yang dikehendaki adalah sama nilainya dengan jumlah statistik-statistik asal selepas ia dibulatkan;

ii Hasil tambah ralat-ralat pembulatan kuasa dua dijadikan paling kecil.

Sebagai contoh:

	Statistik belum dibulatkan	Statistik sudah dibulatkan	Ralat pembulatan kuasa dua
Statistik-statistik individu	4,400	5,000	600^2
	6.300	6,000	300^2
Jumlah	10,700	11,000	450,000

4. PERANTI PAPARAN HARGA STOK

Sebuah firma broker saham ingin membina satu peranti paparan harga saham di luar pejabatnya dengan mengunakan *ultra-bright LED*. Peranti paparan itu adalah setinggi 16 *LED* dan sepanjang 1024 *LED*. Ia terdiri daripada 16 daftar syif 1024-bit yang dimuat dari sebelah kanan peranti paparan. Tugas anda adalah menulis satu program untuk mengawal peranti paparan itu.

Fungsi yang berikut mengalih imej yang berada pada peranti paparan itu satu piksel ke kiri, dan memuatkan jalur paling kanan bagi *LED-LED* itu dengan data yang baru:

Dalam bahasa pengaturcaraan C dan C++:

```
void ShiftAndLoad (unsigned short column);
```

Dalam bahasa pengaturcaraan Java:

```
class DisplayControl {
     //...
     public static void shiftAndLoad (short column);
     //...
}
```

`column`
: ialah satu peta bit (*bitmap*) yang mengandungi keadaan yang dimuatkan ke dalam jalur paling kanan *LED-LED*. Jika satu bit telah diset, *LED* yang berkaitan akan dinyalakan. Jika satu bit diset semula, *LED* yang berkenaan akan dipadamkan. Bit-bit yang lebih bernilai adalah selaras dengan *LED-LED* yang lebih tinggi. Iaitu, bit yang paling tidak bernilai merujuk kepada *LED* yang berada dibawah sekali dan bit yang paling bernilai merujuk kepada *LED* yang paling atas.

Jujukan yang dipaparkan mesti ditakrifkan di dalam satu fail teks yang akan dibaca oleh program kawalan dalam prosess permulaannya.

Melainkan apa yang dinyatakan di bawah, teks di dalam fail teks itu mestilah ditunjukkan di atas peranti paparan seperti yang sedia ada.

Aksara-aksara *new-line* mestilah ditukar kepada aksara-aksara ruang (*space*).

Jika fail teks itu mengandungi satu jujukan dalam bentuk yang berikut, jujukan itu mesti ditukar kepada satu harga stok:

```
<price server=ip_address:port code=stock_code>
```

`ip_address`
: ialah alamat IP bagi pelayan harga stok dalam format dot. Contohnya: 192.0.1.5.

`port`
: ialah nombor port bagi pelayan harga stok. Bahagian '`:`*`port`*' pada alamat pelayan adalah tidak wajib. Jika ia ditinggalkan, port pelayan mesti melalai ke 1845.

`stock_code`
: ialah satu kod alfanumerik 15-aksara yang menyatakan mana satu stok yang harganya perlu dipaparkan. Contohnya, kod stok itu boleh jadi GENTING yang mewakili Genting Berhad.

Sebagai contoh, jujukan

```
<price server=192.0.1.5:1820 code=GENTING>
```

boleh ditukar kepada harga stok 23.70 selepas diproses.

Ruang putih (*white space*) tambahan (iaitu aksara *space*, *tab* dan *new-line*) diantara unsur-unsur leksikal jujukan mestilah diabaikan.

Perkataan-perkataan simpanan ‘price’, ‘server’ dan ‘code’ mestilah diterima dalam huruf besar, huruf kecil atau campuran kedua-duanya.

Fail teks boleh juga mengandungi kod-kod khas yang berikut. Program kawalan mestilah dapat mengenali kod-kod itu dan memaparkan output yang selaras:

Kod Fail Teks	**Output dipaparkan**
`&`	&
`<`	<

Program kawalan itu mestilah menukar aksara-aksara yang kena dipaparkan kepada peta-peta bit dan kemudiannya menyuapkan setiap jalur peta bit kepada fungsi `ShiftAndLoad` dengan kadar pemalar 25 jalur sesaat.

Apabila program pemapar sampai ke hujung jujukan paparan itu, ia mesti memaparkan satu aksara ruang dan kemudian mengulangi jujukan paparan itu dari permulaan.

Program kawalan boleh memperolehi peta bit yang selaras dengan satu aksara tertentu dengan memanggil kaedah `GetBitmap` yang ditunjukkan dibawah.

Dalam bahasa pengaturcaraan C and C++:

```
typedef struct {
     unsigned short           bmColCnt;
     const unsigned short     *bmColData;
} Bitmap_t;

const Bitmap_t *GetBitmap (char ch);
```

Dalam bahasa pengaturcaraan Java:

```
class Bitmap {
     short               bmColCnt;
     short               bmColData[];
}
class DisplayControl {
     //...
     public static Bitmap getBitmap (char ch);
     //...
}
```

`bmColCnt`
ialah bilangan jalur dalam peta bit.

`bmColData`
ialah satu penuding(atau rujukan) kepada satu tatasusunan bagi pembolehubah-pembolehubah integer pendek tanpa tanda, setiap satu mengandungi peta bit untuk satu jalur paparan. Jalur-jalur yang berada di bahagian kiri sesuatu aksara mempunyai alamat-alamat yang rendah dan jalur-jalur di bahagian kanan aksara tersebut beralamat tinggi pada tatasusunan itu.

`ch`
ialah satu aksara dimana peta bitnya dikehendaki.

`return value`
ialah satu penuding (atau rujukan) kepada satu struktur `Bitmap_t` yang mengandungi parameter-parameter peta bit bagi aksara yang dikenalpasti oleh parameter `ch`.

Untuk memperolehi satu harga stok, program kawalan mesti mewujudkan satu sambungan TCP kepada pelayan dan kemudian menghantar satu bingkisan permintaan panjang-tetap kepada pelayan. Format bingkisan permintaan adalah seperti berikut:

Posisi bait	**Data**	**Keterangan**
0 ... 1	1486	Kod permintaan (STOCK_PRICE_REQ)
2 ... 3	16	Kepanjangan bahagian utama bingkisan
4 ... 19	Kod stok	Kod Stok (diakhiri dengan aksara nol)

Pelayan akan membalas dengan bingkisan balas yang berikut:

Posisi bait	**Data**	**Keterangan**
0 ... 1	1487	Kod balas (STOCK_PRICE_RESP)
2 ... 3	4	Kepanjangan bahagian utama bingkisan
4 ... 7	Harga stok	Harga Stok dalam sen

Susunan bait bagi semua integer di dalam bingkisan permintaan and bingkisan balas adalah *big endian*.

Apabila program kawalan dimulakan, ia mesti berikhtiar untuk mendapatkan harga stok bagi setiap stok. Ia tidak perlu menunggu sehingga harga semua stok telah diperolehi sebelum memulakan pemaparan mesej bergerak. Jika harga bagi satu stok belum diperolehi ketika program kawalan memerlukannya, harga itu boleh ditunjukkan sebagai ‘?.??’.

Selepas permintaan awalan harga-harga stok, program kawalan mesti cuba mengemaskinikan harga setiap stok dengan menghantar satu permintaan yang lain kepada pelayan stok dengan secepat mungkin selepas 60 saat berlalu sejak penerimaan harga stok terdahulu.

Sebahagian pelayan harga stok mengembalikan harga bagi lebih daripada satu stok. Program kawalan harus berikhtiar mengurangkan beban rangkaian dengan menghantar permintaan untuk semua harga stok yang boleh didapati daripada satu pelayan tertentu satu demi satu dalam satu sambungan TCP.

Sambungan fizikal kepada sebahagian pelayan harga stok adalah tidak boleh dipercayai. Jika satu pelayan gagal untuk membalas pada waktu muatan awalan harga semua stok atau jika satu harga stok telah ketiggalan masa lebih daripada 5 minit, harga stok itu mesti dipaparkan sebagai ‘?.??’.

Jika satu sambungan ke pelayan tidak dapat diwujudkan atau sambungan ke pelayan terputus pada masa komunikasi, program kawalan mesti membiarkan sambungan itu dan terus mengemaskinikan harga stok yang berikut. Kemudian, ia mesti cuba menyambung semula ke pelayan itu dengan secepat mungkin selepas 60 saat dari masa penemuan kegagalan.