# E-GENTING
# PROGRAMMING COMPETITION 2010

## 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. | Propeller Balancing | 200 |
| 2. | Testing the Customer Query Cache | 150 |
| 3. | Vending Machine Reconciliation Report | 150 |
| 4. | Data Generator | 500 |

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 PROPELLER BALANCING

The pilot of a light aircraft has noticed that the aircraft vibrates significantly in flight. The vibrations have caused damage to a radio and other navigation instruments and need to be corrected. The main issue is that as a consequence of variations in manufacture and wear and tear the propeller blades have slightly different weights. As the propeller turns, it shakes the aircraft. The engineer responsible for maintaining the aircraft has obtained an accelerometer that can be placed on the front of the aircraft engine as shown in Figure 1. He has also fabricated some electronics that convert the amplitude of the vibrations into an electrical signal that can be read with a voltmeter.

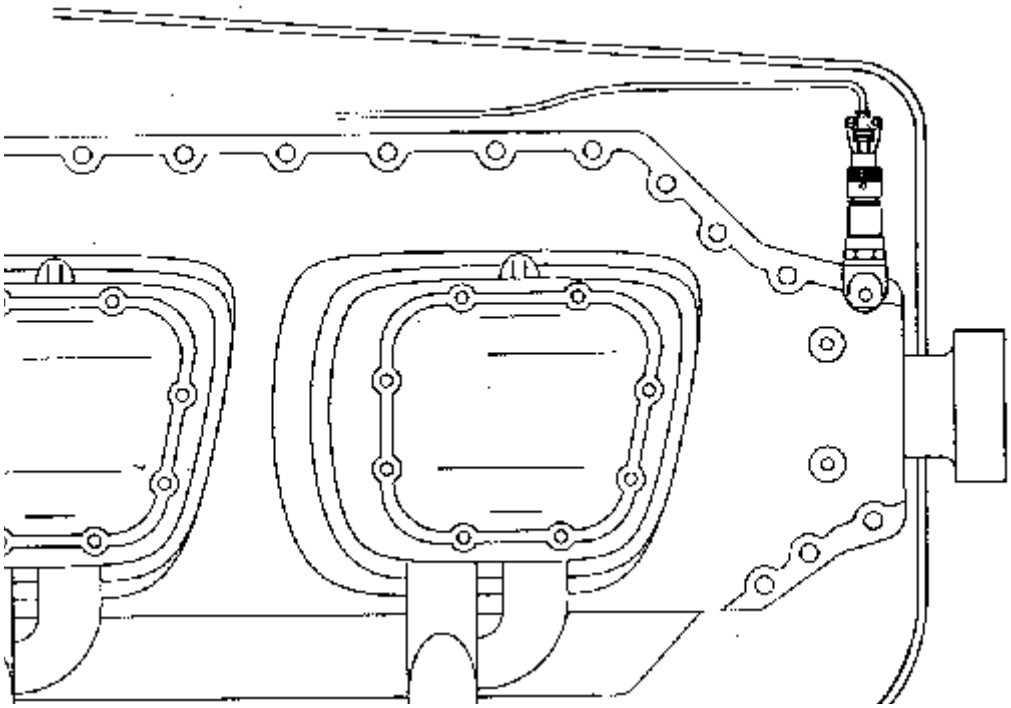

**Figure 1 – Accelerometer Mounting**

The engineer plans to put counterweights on the propeller to correct the vibrations, but he needs to convert the vibration measurements, which are read in volts, into the phase (i.e. angle) and moment (i.e. weight-radius product) of the out-of-balance propeller.

The engineer plans to accomplish this by attaching test weights at different positions on the propeller and measuring the increase or decrease in vibration. If the total vibration increases, the moment of the test weight is aligned with the out-of-balance moment of the propeller. If the vibration decreases, the test weight is opposite to the out-of-balance moment of the propeller.

To cut a long story short, the engineer needs to find values of $V_n$, $V_{pgi}$, $R_p$ and $\Phi_p$ such that $e^2$ is minimised in Equation 1.

**Equation 1 – Sum of Squares of Errors**

$$e^2 = \sum_{i=1}^{n} \left[ V_i - V_n - V_{pgi} \sqrt{(R_p \sin(\Phi_p) + R_i \sin(\Phi_i))^2 + (R_p \cos(\Phi_p) + R_i \cos(\Phi_i))^2} \right]^2$$

Where:

$e^2$ = sum of the errors squared;
$V_n$ = voltage level of background vibration not caused by the out-of-balance propeller;
$V_{pgi}$ = volts per gram-inch (i.e. the relationship between the measured voltage and the out-of-balance moment);
$R_p$ = the magnitude (in gram-inches) of the out-of-balance moment;
$\Phi_p$ = the phase (i.e. angle) of the out-of-balance moment;
$n$ = the number of tests done by the engineer;
$V_i$ = the i-th voltage level read by the engineer during the tests;
$R_i$ = the magnitude (in gram-inches) of the moment of the i-th test weight attached to the propeller;
$\Phi_i$ = the phase (i.e. angle) of the moment of the i-th test weight attached to the propeller.

The value of n is at least 4, but may be greater.

Your task is to write a computer program that searches for values of $V_n$, $V_{pgi}$, $R_p$ and $\Phi_p$ to find the values that minimise $e^2$.

A typical set of sample data is given in Table 1.

**Table 1 – Typical Sample Data**

| $V_i$ (Volts) | $R_i$ (Gram-inches) | $\Phi_i$ (Degrees) |
|---:|---:|---:|
| 0.510 | 0.00 | 0.00 |
| 1.546 | 309.53 | 39.53 |
| 1.141 | 309.53 | 159.54 |
| 0.892 | 309.53 | 279.54 |
| 0.391 | 126.03 | 235.65 |

The value of $V_n$ typically lies between 0.0 and 0.6 Volts. Your program must determine its value to a precision of ±0.001 Volts.

The value of $V_{pgi}$ typically lies between 0.0 and 0.005 Volts per gram-inch. Your program must determine its value to a precision of ±5.0 microvolts per gram-inch.

The value of $R_p$ typically lies between 0.0 and 250.0 gram-inches. Your program must determine its value to a precision of ±0.1 gram-inches.

The value of $\Phi_p$ will lie between 0 and 360 degrees. Your program must determine its value to a precision of ±0.1 degree.

Some preliminary research indicates that a naïve design that simply tests each possible value of $V_n$, $V_{pgi}$, $R_p$ and $\Phi_p$ to their required precisions will take 78 days to run. This is unacceptably slow. Your program should produce the correct answer in less than 5 minutes.

# 2 TESTING THE CUSTOMER QUERY CACHE

A marketing system used by the Genting Group, uses a cache, known as the Customer Query Cache or CQC, to quickly look up the customer identifiers that match all or part of a customer's name. The CQC is implemented as a class that has the declarations in Figure 2 and Figure 3.

The cache groups the customer data into large blocks to make best use of the virtual memory swapping characteristics of the underlying operating system and hardware. When a block becomes full, a new block is allocated and added to a block chain and when a block is emptied, the block is removed from the block chain and returned to the free list.

Several users of the marketing system have reported that occasionally, when they add a customer to the database, and then search on the customer's name, the customer does not appear on the output extracted from the cache, but tests using small amounts of data have failed to find any defect in the cache.

Your task is to write a program that randomly inserts and removes records from the cache and then periodically verifies that the contents of the cache is correct.

The program must pseudo-randomly decide to insert a row, remove an arbitrarily selected row, or verify the contents of the cache in a cycle that, on the average, progressively increases the number of customer records stored in the cache. The cache in the marketing system stores approximately 2 million customer records.

The program must pseudo-randomly select the customer identifiers and names to be inserted. New customer identifiers must not duplicate customer identifiers already in the cache. Customer names should contain a pseudo-random selection of the letters A to Z and the space character. The customer names should be between 1 and 40 characters long.

The program must verify that the customer identifiers and names stored in the cache are correct approximately once for every 1,000 insert or delete operations.

The CQC declarations are contained in the C++ header file "Cqc.h".

CqcNextCust does not return the customer data records in any particular order.

```
// Customer Data Structure

struct Cust_t {
     long      custId;   // Customer identifier
     string    custName; // Customer name
     Cust_t () { /*...*/ }
                         // Default constructor
     Cust_t (const Cust_t &cust) { /*...*/ }
                         // Copy constructor
     Cust_t &operator = (const Cust_t &cust) { /*...*/ }
                         // Assignment operator
};

// Customer Query Cache Class

class Cqc_c {
     // ...
public:
     void CqcInsert (const Cust_t *cust);
                         // Insert a customer record into
                         // the cache.
     void CqcRemove (long custId);
                         // Remove the customer record with
                         // customer identifier 'custId'
                         // from the cache.
     void CqcSelect (const char *key);
                         // Select the customers with names
                         // that match 'key'.  'key' may
                         // contain wild cards.  For
                         // example "*" selects all
                         // customers.
     const Cust_t *CqcNextCust ();
                         // Return a pointer to the next
                         // selected customer data
                         // structure, return zero at
                         // end-of-file
};
```

**Figure 2 – Customer Query Cache in C++**

```
// Customer Data Structure

public class Cust {
     int       custId;   // Customer identifier
     String    custName; // Customer name
}

// Customer Query Cache Class

public class Cqc {
     void cqcInsert (Cust cust) { /*...*/ }
                          // Insert a customer record into
                          // the cache.
     void cqcRemove (int custId) { /*...*/ }
                          // Remove the customer record with
                          // customer identifier 'custId'
                          // from the cache.
     void cqcSelect (String key) { /*...*/ }
                          // Select the customers with names
                          // that match 'key'.  'key' may
                          // contain wild cards.  For
                          // example "*" selects all
                          // customers.
     Cust cqcNextCust () { /*...*/ }
                          // Return a reference to the next
                          // selected customer data
                          // structure, return null at
                          // end-of-file
}
```

**Figure 3 – Customer Query Cache in Java**

# 3 VENDING MACHINE RECONCILIATION REPORT

The Coinless Vending Company operates vending machines in airports and railway and bus stations. The vending machines sell drinks and other snacks to people waiting for planes, trains and buses. Coinless distinguishes its vending machines by a card-based coinless system that lets customers hold their credit on a card instead of taking their change in coins.

Each of Coinless's vending machines is equipped with a device known as a machine interface that has a card reader, a display and an interface to a central computer as shown in Figure 4.

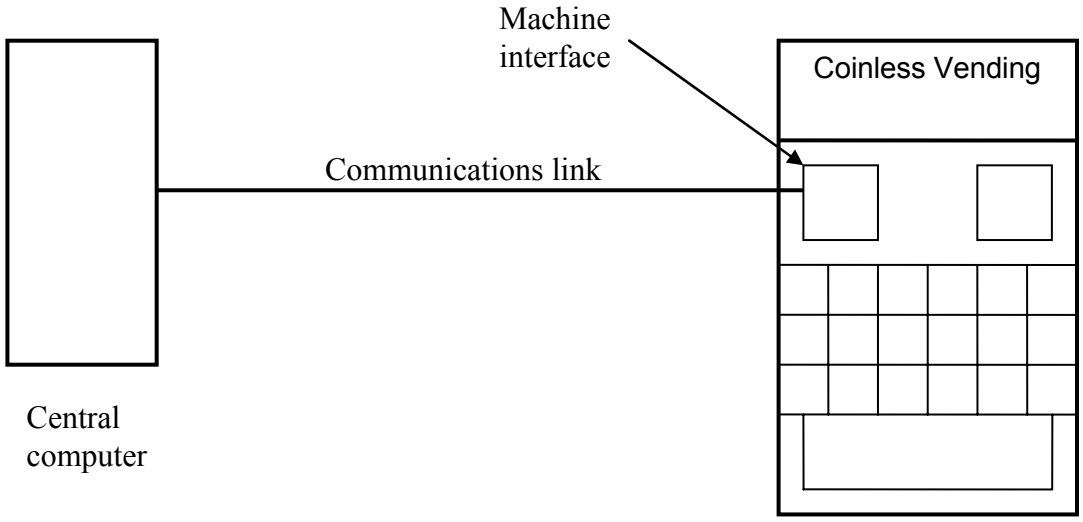


**Figure 4 – Vending Machine Communications**

When a customer inserts his card into the machine interface's card reader, the machine interface sends the card number to the central computer, which sends the customer's balance along the communications link to the machine interface. The machine interface then credits the customer's balance to the credit meter of the vending machine.

Once the customer's balance is credited to the credit meter of the vending machine, the customer may use the controls of the vending machine to select his purchase. He may also insert banknotes into the bill validator on the vending machine, which will increase his credit.

The machine interface has a button labelled 'collect card', that the customer presses to transfer any residual credit back to his account on the central computer and get his card back.

The vending machine has meters that total the following statistics:

1. money transferred in from the machine interface (MTI);
2. money transferred out to the machine interface (MTO);
3. bills inserted by the customer (BI);
4. sales of each item (S).

These meters are read electronically each day at around 6am when the money is collected from the bill validator in the vending machine.

Coinless's central system produces customer statements that the customers can view over the Internet. Customer account days are based on calendar days and roll over at midnight. A transaction occurring at say 23:59 on 22 October will be posted to the customer's account for 22 October, whereas a transaction occurring at 00:01 on 23 October will be posted to the customer's account for 23 October. In contrast, because of the arrangements for collecting money from the vending machines, a vending machine

transaction occurring at 03:00 on 23 October will be posted to the machine account for 22 October. Only when the money is collected from the vending machines at around 6am will transactions be posted to the machine account for the new day.

Additionally, the communications link between the machine interface and the central computer is not 100% reliable. Occasionally, when the machine interface attempts to transfer the customer's money back to the customer's account on the central computer, it is unable to do so. In this situation the machine interface transfers the money back to the credit meter on the vending machine. This results in the customer's credit being added to the MTO meter when the money is initially transferred from the credit meter of the vending machine to the machine interface and then being added to the MTI meter when the money is transferred back to the credit meter of the vending machine after the machine interface encounters the communications fault. Neither of these transfers gets reflected on the customer's account.

An accountant at Cashless has compared the amounts transferred to and from vending machines in the customer statements with the amounts transferred in and out of the vending machines as recorded by the meters on the vending machines and has found that they are not equal.

Your task is to write a reporting program that produces a Vending Machine Reconciliation Report that explains the differences between the amounts transferred from the customer accounts and the amount recorded by the MTI meter of the vending machine. There are also reconciliation errors in relation to the MTO meter, but these are beyond the scope of the current project.

The reporting program must accept the following parameters:

1. from-date,
2. to-date.

The Vending Machine Reconciliation Report must contain the following information:

(a) the total value of amounts transferred from customer accounts to machine interfaces between the from-date and to-date inclusive as shown in customer statements;
(b) the value of transfers from customer accounts to machine interfaces that were recorded in the from-date on the customer account and the day before the from-date on the MTI meter of the vending machine;
(c) the value of transfers from customer accounts to machine interfaces that were recorded in the day after the to-date on the customer account, but on the to-date on the MTI meter of the vending machine;
(d) the value of reversed transfers from the vending machine to customer accounts that were recorded on the MTI meter, but not on any customer account;
(e) sub-total of (a) – (b) + (c) + (d);
(f) the total value of amounts recorded on the MTI meter;
(g) if (e) ≠ (f), the title 'Meter mismatch' and the amount (e) – (f).

The Vending Machine Reconciliation Report should be laid out in accordance with the example in Figure 5.

```
VENDING MACHINE RECONCILIATION REPORT
FOR DD-MM-YY TO DD-MM-YY

Account-to-machine transfers
     As shown in customer accounts              10,872.50
     Less transactions occurring in the              5.20
     previous machine day
     Plus transactions logged in the next            8.50
     customer day that occurred in the
     current machine day
     Plus reversed transfers                        14.00
                                                ---------
     Sub-total                                  10,889.80
     Vending machine metered value              10,889.60
                                                ---------
     Meter mismatch                                  0.20
```

**Figure 5 – Vending Machine Reconciliation Report Example**

The reporting program must extract the information it needs to produce the Vending Machine Reconciliation Report from the Cashless System database. The database contains the following tables that are relevant to the report:

```
create table custTx (
     custId         integer not null,
     custDate       date not null,
     txNo           integer not null
);
create table vendTx (
     machId         integer not null,
     machDate       date not null,
     txNo           integer not null
);
create table txData (
     txNo           integer not null,
     txTime         timestamp not null,
     txType         smallint not null,
     txValue        double precision not null
);
create table vendMeters (
     machId         integer not null,
     machDate       date not null,
     meterId        smallint not null,
     meteredValue   double precision not null
);
```

`custTx`
: is a table that contains one row for each transaction shown on the customer accounts.

`custTx.custId`
: is the customer's customer identifier.

`custTx.custDate`
: is the date of the transaction in the customer's statements.

`custTx.txNo`
: is a transaction number that uniquely identifies the transaction. `custTx` and `txData` may be joined on `txNo` to obtain extended information about the transaction.

`vendTx`
: is a table that contains one row for each transaction that affects the meters on the vending machine.

`vendTx.machId`
: is a machine identifier that uniquely identifies the vending machine.

`vendTx.machDate`
: is the date of the transaction in the machine account, which rolls over at approximately 6am.

`vendTx.txNo`
: is a transaction number that uniquely identifies the transaction. `vendTx` and `txData` may be joined on `txNo` to obtain extended information about the transaction.

`txData`
: is a table that contains one row for each transaction that affects either a customer account or a machine account.

`txData.txNo`
: is a transaction number the uniquely identifies the transaction.

`txData.txTime`
: is the real time of the transaction.

`txData.txType`
: is a transaction type code. The transaction type codes are shown in Table 2.

`txData.txValue`
: is the transaction value in cents. For example, $12.34 is recorded as 1234.0.

`vendMeters`
: is a table that contains one row for each meter for each machine day.

`vendMeters.machId`
: is a machine identifier that uniquely identifies the vending machine.

`vendMeters.machDate`
: is the machine account date.

`vendMeters.meterId`

is a meter identifier that uniquely identifies the meter. The meter identifiers are listed in Table 3.

`vendMeters.meteredValue`

is the difference between the closing and opening meter reading in cents, after taking into account issues such as meter rollovers.

**Table 2 – Transaction Type codes**

| Code | Meaning |
|---|---|
| 1 | Transfers from a customer's account to a machine interface |
| 2 | Transfers from a machine interface to a customer's account |
| 3 | Insertion of a banknote into the vending machine |
| 4 | Sale |
| 5 | Attempted transfer from the machine interface to the central computer that could not be completed and was reversed by the machine interface sending the money back to the credit meter on the vending machine. |

**Table 3 – Meter Identifiers**

| Code | Meaning |
|---|---|
| 0 | Money transferred in from the machine interface (MTI) |
| 1 | Money transferred out to the machine interface (MTO) |
| 2 | Bills inserted by the customer (BI) |
| 3 | Sales of each item (S) |

The Cashless System inserts rows in the database as shown in Table 4.

**Table 4 – Tables the Transactions Update**

| Transaction | Tables into which rows are inserted |
|---|---|
| Transfers from a customer's account to a machine interface | `custTx, vendTx, txData` |
| Transfers from a machine interface to a customer's account | `custTx, vendTx, txData` |
| Insertion of a banknote into the vending machine | `custTx, vendTx, txData` |
| Sale | `custTx, vendTx, txData` |
| Attempted transfer from the machine interface to the central computer that could not be completed and was reversed by the machine interface sending the money back to the credit meter on the vending machine. | `vendTx, txData` |
| Clearing the bank notes from the bill validator at the end of each machine day at around 6am. | `vendMeters` |

Programmers writing in C or C++ may make use of the date conversion functions described in Figure 6. Java and C-sharp programmers should use the date conversion functions provided by the Java and C-sharp language libraries.

```
#include "dtime.h"
                            // Include declarations of double time
                            // functions.
typedef double Dtime_t;
                            // Double time data type.  Dates and times are
                            // stored in seconds since 00:00 on
                            // 1 January 1970.  Dates are stored as seconds
                            // since 00:00 on 1 January 1970 until 00:00 on
                            // the stored date.
static const Dtime_t NULL_DTIME = -(65536.0*65536.0*65536.0*65536.0);
                            // Reserved double time value (before the big
                            // bang) used to represent an invalid time.
Dtime_t PackDtime (int yr, int mo, int dy, int hr, int mi, int se);
                            // Pack a double time value from its
                            // components.  yr is a 4-digit year (e.g.
                            // 2008).  mo is the calendar month number
                            // (e.g. 1 for January).  dy is the day of the
                            // month (e.g. 1 for the first of the month).
                            // hr, mi and se are the hour, minute and
                            // second components of a 24-hour clock.
                            // PackDtime returns NULL_DTIME if the
                            // components do not represent a valid date
                            // and time.
void UnpackDtime (Dtime_t dtime, int *yr, int *mo, int *dy,
       int *hr, int *mi, int *se);
                            // Unpack a double time value into its
                            // components.
Dtime_t DtimeFromSql (const char *sqlTime);
                            // Convert an SQL date or timestamp string into
                            // double time format.  DtimeFromSql returns
                            // NULL_DTIME if the SQL date or timestamp
                            // string is invalid.
char *SqlDateFromDtime (char *sqlDate, Dtime_t dtime);
                            // Load an SQL date string from a double time
                            // value.  SqlDateFromDtime returns sqlDate.
char *SqlTimestampFromDtime (char *sqlTimestamp, Dtime_t dtime);
                            // Load an SQL timestamp string from a double
                            // time value.  SqlTimestampFromDtime returns
                            // sqlTimestamp.
Dtime_t DtimeFromUserDate (const char *userDate);
                            // Convert a user date in DD-MM-YY format into
                            // the double time format.
char *UserDateFromDtime (char *userDate, Dtime_t dtime);
                            // Convert the date components of a double time
                            // value into a user date string in DD-MM-YY
                            // format. UserDateFromDtime returns userDate.
char *UserTimeFromDtime (char *userTime, Dtime_t dtime);
                            // Convert the time components of a double time
                            // value into a user time string in HH:MM:SS
                            // format. UserTimeFromDtime returns userTime.
```

**Figure 6 – Double Time Functions**

# 4 DATA GENERATOR

Write a data generator for the database described in Question 3.

The data generator must generate data for the period 1 January 2010 (the 'from-date') to 30 June 2010 (the 'to-date').

The data generator must operate in a virtual time domain that starts at 00:00 on the from-date and continues forward until the virtual time reaches 24:00 on the to-date. The data generator must identify the next thing that needs to be done and then advance the virtual time to that time, do the thing, and then again identify the next thing that needs to be done.

The data generator must emulate the behaviour of 100 customers. The emulator for each customer must do the following concurrently with the emulators for all the other customers:

1. Pseudo-randomly select a starting balance for the customer between $10 and $100.
2. While the virtual time is less than 24:00 on the to-date:
    a. Choose a next transaction time between the current virtual time and 48 hours in the future with the time being 5 times more likely to be between 8am and 8pm than at other times.
    b. Wait until the virtual time is the next transaction time.
    c. Pseudo-randomly choose a vending machine from among 20 alternatives.
    d. If the vending machine is in use:
        i. Wait in a queue until the machine is free.
    e. Insert card into the vending machine.
    f. Wait for the virtual time to advance by a pseudo-random period between 5 and 15 seconds.
    g. Choose a purchase value between $2 and $20.
    h. If the customer's balance is insufficient to pay for the purchase:
        i. Choose a bill-in value between the minimum additional balance needed to complete the purchase and $100.
        ii. Emulate the customer inserting bills into the vending machine up to the selected bill-in value.
        iii. Wait for the virtual time to advance by a pseudo-random period between 5 and 15 seconds.
    i. Emulate a sale of value equal to the purchase value.
    j. Wait for the virtual time to advance by a pseudo-random period between 5 and 15 seconds.
    k. Do:
        i. With a likelihood of 1 in 200:
            1. Emulate an attempt to transfer the customer's balance back to the central computer that encounters a communication failure;
        ii. Else:
            1. Emulate a successful transfer of the customer's balance back to the central computer;

iii. End if.
iv. Wait for the virtual time to advance by a pseudo-random period between 5 and 15 seconds.

l. Until emulation of a successful transfer of the customer's balance back to the central computer.
m. Release the vending machine to any other process waiting for it.

3. End while.

Concurrently with emulating the actions of the customers, the database generator must also emulate the actions of vending machine attendants. There must be one vending machine attendant emulator for each vending machine. The vending machine attendant emulator must:

1. Loop:
    a. Pseudo-randomly choose a day roll over time between 05:00 and 07:00 on the next day.
    b. If the day roll over time is greater than or equal to 24:00 on the to-date, exit the loop.
    c. Wait for the virtual time to advance to the day roll over time.
    d. If the vending machine is in use:
        i. Wait in a queue until the machine is free.
    e. Emulate the clearing of the vending machine and the insertion of a new vendMeters row into the database.
    f. Wait for the virtual time to advance by a pseudo-random period between 5 and 15 seconds.
    g. Release the vending machine to any other process waiting for it.
2. End Loop.

# PERTANDINGAN PENGATURCARAAN E-GENTING 2010

## Arahan-arahan am:

1. Jawab satu atau lebih soalan yang diberikan.
2. Pertandingan ini adalah satu ujian di mana peserta-peserta dibenarkan untuk merujuk kepada buku-buku atau bahan-bahan rujukan.
3. Masa yang diperuntukan untuk pertandingan ini adalah 8 jam.
4. Perbincangan sesama peserta tidak dibenarkan.
5. Untuk menerima kredit dalam menjawab sesuatu soalan, jawapan anda mestilah merupakan penyelesaian yang munasabah. Penyelesaian yang munasabah ialah jawapan yang menyelesaikan masalah tersebut atau jawapan yang mungkin menyelesaikan masalah tersebut dengan sedikit usaha tambahan.
6. Sekiranya jawapan anda merupakan penyelesaian yang munasabah, anda akan menerima kredit untuk hasilan methodikal seperti gambar rajah aliran data, gambar rajah peralihan keadaan, jadual dan sebagainya.
7. Jumlah markah anda adalah jumlah kredit yang anda perolehi bagi setiap penyelesaian yang munasabah.
8. Program-program anda akan dinilai berdasarkan kepada betapa mudah ianya 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. | Pengimbangan Kipas Enjin | 200 |
| 2. | Menguji Cache Pencarian Pelanggan | 150 |
| 3. | Laporan Penyelarasan Mesin Jualan | 150 |
| 4. | Penjana Data | 500 |

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’ (mesti), ‘must not’ (tidak mesti), ‘required’ (wajib), ‘should’ (harus), ‘should not’ (tidak harus), dan ‘may’ (boleh) adalah ditafsirkan seperti diterangkan dalam RFC 2119[2].

13. Anda TIDAK perlu menjawab semua soalan.

---

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

# 1 PENGIMBANGAN KIPAS ENJIN

Juruterbang sebuah kapal terbang kecil mendapati bahawa kapal terbang tersebut bergetar dengan ketara ketika dalam penerbangan. Getaran tersebut telah merosakkan sebuah radio dan alat-alat pengemudian yang lain, dan perlu dibetulkan. Perbezaan dalam pembinaan dan kehausan pada kipas enjin telah mengakibatkan perbezaan berat antara bilah-bilah kipas enjin. Ini menyebabkan goncangan kepada kapal terbang apabila kipas enjin berputar. Jurutera yang bertanggungjawab mengendali kapal terbang ini telah memperolehi satu meter pecutan (accelerometer) untuk dipasang di depan kapal terbang seperti dalam Figure 1. Dia juga telah membina alat elektronik untuk menukar kekuatan getaran kepada isyarat elektrik yang boleh dibaca dengan meter volt (voltmeter).

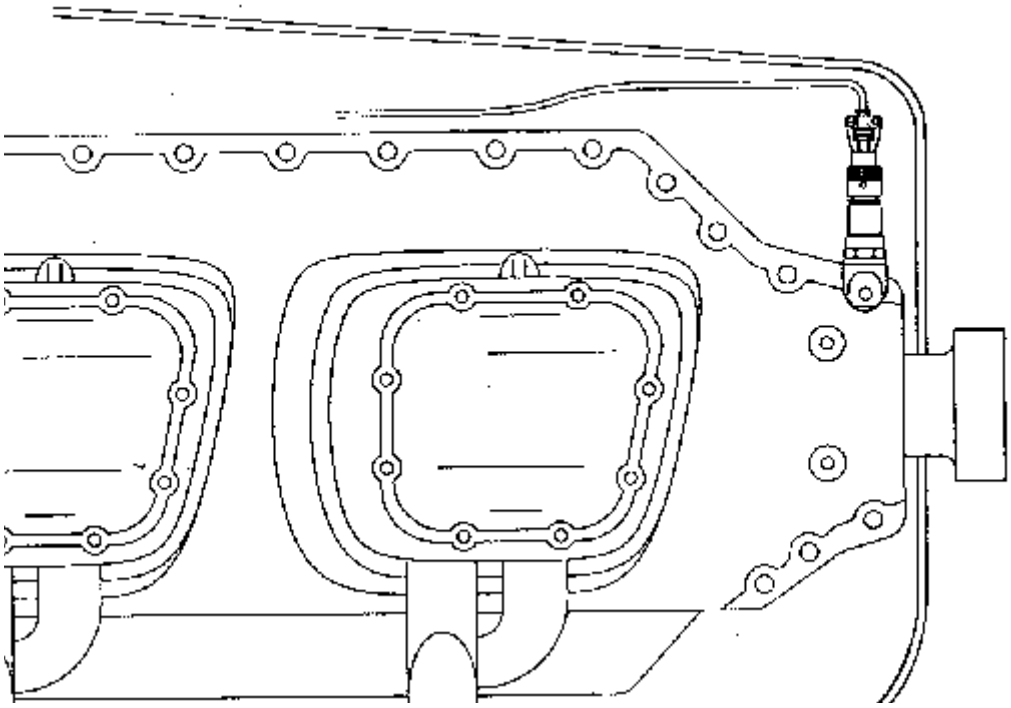

**Figure 1 – Penempatan Accelerometer**

Jurutera tersebut bercadang untuk meletak beban imbangan pada kipas enjin untuk membetulkan getaran, tetapi sebelum itu dia perlu menukar ukuran getaran yang dibaca dalam unit volt kepada fasa (iaitu sudut) dan momen (iaitu hasil darab berat dan jejari) bagi kipas enjin yang tidak seimbang itu.

Jurutera tersebut cuba menempatkan pemberat pada kedudukan yang berbeza atas kipas enjin dan kemudian mengukur sama ada getaran bertambah atau berkurang. Jika getaran bertambah, maka momen pemberat tersebut adalah sejajar dengan momen kipas enjin yang tidak seimbang. Jika getaran berkurang, maka momen pemberat adalah bertentangan dengan momen kipas enjin yang tidak seimbang.

Secara kesimpulan, jurutera tersebut perlu mengenalpasti nilai-nilai bagi $V_n$, $V_{pgi}$, $R_p$ dan $\Phi_p$ supaya meminimumkan nilai $e^2$ dalam Equation 1.

**Equation 1 – Jumlah Kesilapan Dikuasa-duakan**

$$e^2 = \sum_{i=1}^{n} \left[ V_i - V_n - V_{pgi} \sqrt{(R_p \sin(\Phi_p) + R_i \sin(\Phi_i))^2 + (R_p \cos(\Phi_p) + R_i \cos(\Phi_i))^2} \right]^2$$

Di mana:

$e^2$ = jumlah kesilapan dikuasa-duakan;

$V_n$ = tahap nilai volt getaran latar belakang yang bukan disebabkan oleh kipas enjin yang tidak seimbang;

$V_{pgi}$ = volt per gram-inci (iaitu hubungan di antara nilai volt yang diukur dengan momen tidak seimbang);

$R_p$ = magnitud (dalam gram-inci) bagi momen tidak seimbang;

$\Phi_p$ = fasa (iaitu sudut) bagi momen tidak seimbang itu;

$n$ = bilangan ujian yang dijalankan oleh jurutera tersebut;

$V_i$ = tahap nilai volt ke-i yang dibaca oleh jurutera semasa ujian;

$R_i$ = magnitud (dalam gram-inci) bagi momen pemberat ke-i yang dipasang pada kipas enjin;

$\Phi_i$ = fasa (iaitu sudut) bagi momen pemberat ke-i yang dipasang pada kipas enjin.

Nilai untuk n adalah sekurang-kurangnya 4, tetapi boleh lebih besar daripada itu.

Tugas anda ialah menulis satu aturcara komputer yang mengenalpasti nilai-nilai $V_n$, $V_{pgi}$, $R_p$ dan $\Phi_p$ supaya meminimumkan nilai $e^2$.

Satu contoh sampel data diberi dalam Table 1.

**Table 1 – Sampel Data Tipikal**

| $V_i$ (Volt) | $R_i$ (Gram-inci) | $\Phi_i$ (darjah) |
|---:|---:|---:|
| 0.510 | 0.00 | 0.00 |
| 1.546 | 309.53 | 39.53 |
| 1.141 | 309.53 | 159.54 |
| 0.892 | 309.53 | 279.54 |
| 0.391 | 126.03 | 235.65 |

Nilai $V_n$ biasanya berada di antara 0.0 dan 0.6 Volt. Aturcara anda mesti mengenalpasti nilainya sehingga kepersisan ±0.001 Volts.

Nilai $V_{pgi}$ biasanya berada di antara 0.0 dan 0.005 Volt per gram-inci. Aturcara anda mesti mengenalpasti nilainya sehingga kepersisan ±5.0 microvolts per gram-inci.

Nilai $R_p$ biasanya berada di antara 0.0 dan 250.0 gram-inci. Aturcara anda mesti mengenalpasti nilainya sehingga kepersisan ±0.1 gram-inci.

Nilai $\Phi_p$ adalah di antara 0 dan 360 darjah. Aturcara anda mesti mengenalpasti nilainya sehingga kepersisan ±0.1 darjah.

Kajian awal menunjukkan bahawa penyelesaian mudah yang cuba menguji setiap nilai yang berkemungkinan bagi $V_n$, $V_{pgi}$, $R_p$ and $\Phi_p$ sehingga ke kepersisan yang dikehendaki memerlukan 78 hari untuk dilaksanakan. Ini adalah terlampau lambat. Aturcara anda harus memberikan jawapan yang betul dalam masa kurang daripada 5 minit.

# 2 MENGUJI CACHE PENCARIAN PELANGGAN

Satu sistem pemasaran yang digunakan oleh Genting Group menggunakan satu cache, yang dikenali sebagai Cache Pencarian Pelanggan atau CQC, untuk mengenalpasti nombor identiti pelanggan yang mempunyai semua atau sebahagian daripada nama pelanggan dengan cepat. CQC dibina sebagai satu kelas yang mempunyai pengisytiharan seperti dalam Figure 2 dan Figure 3.

Cache tersebut mengumpul data pelanggan dalam blok-blok besar untuk menggunakan ciri penukaran ingatan maya yang dibekalkan oleh sistem operasi dan perkakasan komputer secara optimum. Apabila sesuatu blok telah penuh, satu blok baru akan diperuntukan dan ditambah kepada rantai blok. Apabila sesuatu blok dikosongkan, blok tersebut disingkirkan daripada rantai blok dan dikembalikan kepada senarai bebas.

Beberapa pengguna sistem pemasaran melaporkan bahawa kadang kala selepas mereka menambah rekod pelanggan ke dalam pangkalan data dan kemudian mencari nama pelanggan tersebut, rekod pelanggan tersebut tidak ditunjukkan dalam hasil pencarian yang diperolehi dari cache. Ujian yang menggunakan sedikit data telah gagal mengesan sebarang kecacatan dalam cache tersebut.

Tugas anda ialah untuk menulis satu aturcara yang menambah dan menyingkir rekod dari cache secara rawak dan mengesahkan kandungan cache adalah tepat secara berkala.

Aturcara tersebut mesti menentukan secara pseudo rawak sama ada untuk menambah satu baris, menyingkir sesuatu baris, atau mengesahkan kandungan cache dengan ulangan di mana secara purata, bilangan rekod pelanggan dalam cache bertambah secara progresif. Cache dalam sistem pemasaran menyimpan lebih kurang 2 juta rekod pelanggan.

Aturcara tersebut mesti memilih nombor identiti dan nama pelanggan yang perlu dimasukkan secara pseudo rawak. Nombor identiti pelanggan baru mestilah tidak berulang dengan nombor identiti pelanggan yang telah disimpan dalam cache. Nama pelanggan harus mengandungi huruf pilihan pseudo rawak dari A hingga Z dan juga aksara ruang. Nama pelanggan harus mempunyai panjang di antara 1 hingga 40 aksara.

Aturcara tersebut mesti melaksanakan pengesahan bahawa nombor identiti dan nama pelanggan yang disimpan dalam cache adalah betul dalam kekerapan lebih kurang sekali dalam setiap 1,000 operasi tambah atau singkir.

Pengisytiharan CQC dibekal dalam fail C++ “Cqc.h”.

CqcNextCust tidak mengembalikan rekod pelanggan mengikut mana-mana susunan.

```
// Customer Data Structure

struct Cust_t {
     long      custId;   // Customer identifier
     string    custName; // Customer name
     Cust_t () { /*...*/ }
                         // Default constructor
     Cust_t (const Cust_t &cust) { /*...*/ }
                         // Copy constructor
     Cust_t &operator = (const Cust_t &cust) { /*...*/ }
                         // Assignment operator
};

// Customer Query Cache Class

class Cqc_c {
     // ...
public:
     void CqcInsert (const Cust_t *cust);
                         // Insert a customer record into
                         // the cache.
     void CqcRemove (long custId);
                         // Remove the customer record with
                         // customer identifier 'custId'
                         // from the cache.
     void CqcSelect (const char *key);
                         // Select the customers with names
                         // that match 'key'.  'key' may
                         // contain wild cards.  For
                         // example "*" selects all
                         // customers.
     const Cust_t *CqcNextCust ();
                         // Return a pointer to the next
                         // selected customer data
                         // structure, return zero at
                         // end-of-file
};
```

**Figure 2 – Cache Pencarian Pelanggan dalam C++**

```
// Customer Data Structure

public class Cust {
     int       custId;   // Customer identifier
     String    custName; // Customer name
}

// Customer Query Cache Class

public class Cqc {
     void cqcInsert (Cust cust) { /*...*/ }
                          // Insert a customer record into
                          // the cache.
     void cqcRemove (int custId) { /*...*/ }
                          // Remove the customer record with
                          // customer identifier 'custId'
                          // from the cache.
     void cqcSelect (String key) { /*...*/ }
                          // Select the customers with names
                          // that match 'key'.  'key' may
                          // contain wild cards.  For
                          // example "*" selects all
                          // customers.
     Cust cqcNextCust () { /*...*/ }
                          // Return a reference to the next
                          // selected customer data
                          // structure, return null at
                          // end-of-file
}
```

**Figure 3 – Cache Pencarian Pelanggan dalam Java**

# 3 LAPORAN PENYELARASAN MESIN JUALAN

Syarikat Coinless Vending mengendalikan mesin-mesin jualan di lapangan terbang dan stesen-stesen keretapi dan bas. Mesin-mesin tersebut menjual minuman dan snek kepada penumpang-penumpang yang sedang menunggu kapal terbang, keretapi atau bas. Ciri utama mesin-mesin syarikat tersebut ialah penggunaan sistem bayaran tanpa syiling berasaskan kad yang membolehkan pelanggan-pelanggan menyimpan kredit dalam kad dan tidak perlu menggunakan syiling.

Setiap mesin jualan dipasang dengan satu alat yang dikenali sebagai antaramuka mesin yang mempunyai satu pembaca kad, satu paparan dan satu antaramuka kepada satu komputer pusat seperti ditunjukkan dalam Figure 4.

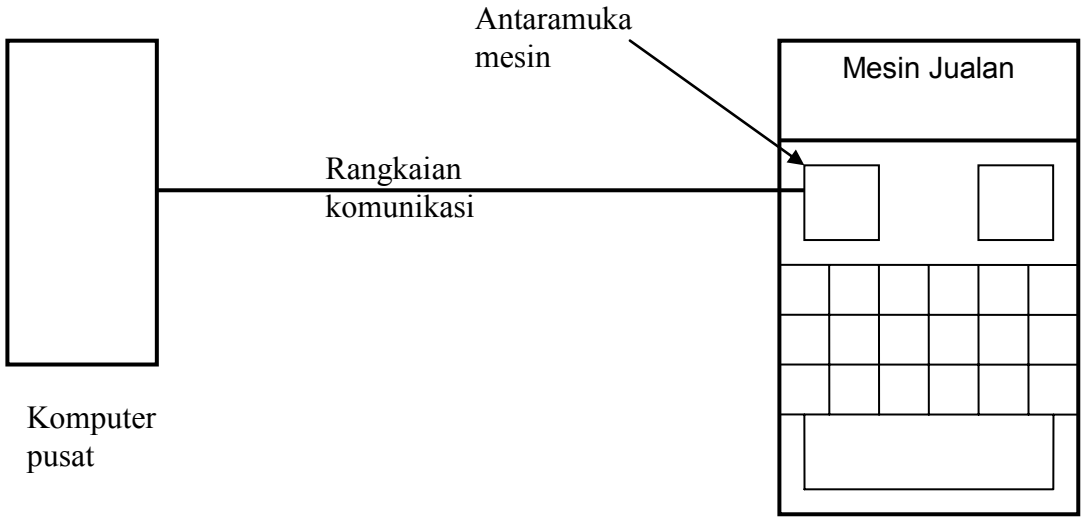


**Figure 4 – Komunikasi Mesin Jualan**

Apabila pelanggan memasukkan kad ke dalam pembaca kad pada antaramuka mesin, antaramuka mesin tersebut menghantar nombor kad kepada komputer pusat, yang kemudiannya mengembalikan baki kredit pelanggan melalui rangkaian komunikasi yang berhubung dengan antaramuka mesin. Antaramuka mesin seterusnya mengkreditkan nilai baki pelanggan kepada meter kredit mesin jualan tersebut.

Setelah baki pelanggan dikreditkan ke meter kredit mesin jualan tersebut, pelanggan boleh menggunakan unit kawalan pada mesin jualan untuk membuat pembelian. Pelanggan juga boleh masukkan wang kertas ke dalam pengesah wang kertas pada mesin jualan, di mana ia akan menambah kredit pelanggan.

Antaramuka mesin tersebut mempunyai satu butang yang dilabel sebagai ‘kutip kad’, di mana pelanggan boleh menekan butang tersebut untuk mengembalikan baki kredit kepada komputer pusat dan mengambil balik kadnya.

Mesin jualan tersebut mempunyai meter-meter yang menjumlahkan statistik berikut:

1. wang dipindah masuk dari antaramuka mesin (MTI);
2. wang dipindah keluar ke antaramuka mesin (MTO);
3. wang kertas dimasukkan oleh pelanggan (BI);
4. jualan bagi setiap barang (S).

Meter-meter ini dibaca secara elektronik setiap hari pada kira-kira jam 6 pagi apabila wang kertas dikutip dari pengesah wang kertas dalam mesin jualan.

Sistem pusat syarikat tersebut menghasilkan penyata pelanggan yang boleh dicapai oleh pelanggan melalui Internet. Hari akaun pelanggan adalah berdasarkan hari kalendar dan dimaju ke depan pada tengah malam. Satu urusniaga berlaku pada contohnya 23:59 22 Oktober akan direkod pada akaun pelanggan bagi 22 Oktober, sebaliknya satu urusniaga yang berlaku pada 00:01 23 Oktober akan direkod pada akaun pelanggan bagi 23

Oktober. Sebagai perbandingan, disebabkan oleh masa pengutipan wang dari mesin jualan, sesuatu urusniaga mesin jualan yang berlaku pada 03:00 23 Oktober akan direkod pada akaun mesin bagi 22 Oktober. Hanya setelah wang dikutip dari mesin-mesin jualan pada kira-kira jam 6 pagi, urusniaga selanjutnya akan direkod pada akaun mesin bagi hari yang baru.

Tambahan pula, rangkaian komunikasi di antara antaramuka mesin dan komputer pusat bukanlah 100% stabil. Kadang-kadang, percubaan antaramuka mesin untuk mengembalikan wang pelanggan ke akaun pelanggan di komputer pusat tidak berjaya. Dalam keadaan sebegini, antaramuka mesin memindah wang tersebut kembali ke meter kredit di mesin jualan tersebut. Ini menyebabkan kredit pelanggan ditambah ke meter MTO pada permulaan di mana wang dipindah dari meter kredit mesin jualan ke antaramuka mesin dan kemudian ditambah ke meter MTI apabila wang dipindah balik ke meter kredit mesin jualan selepas antaramuka mesin menghadapi kegagalan komunikasi. Kedua-dua pindahan ini tidak dapat direkod pada akaun pelanggan.

Seorang akauntan di syarikat tersebut telah membandingkan jumlah yang dipindah masuk dan keluar dari mesin jualan yang dicatat pada penyata-penyata pelanggan dengan jumlah yang dipindah masuk dan keluar dari mesin jualan yang direkod pada meter-meter mesin jualan. Beliau mendapati kedua-dua jumlah adalah tidak sama.

Tugas anda ialah untuk menulis satu aturcara yang menghasilkan satu Laporan Penyelarasan Mesin Jualan yang menerangkan perbezaan antara jumlah yang dipindah dari akaun pelanggan dan jumlah yang direkodkan oleh meter MTI pada mesin jualan. Ralat dalam penyelarasan juga berlaku untuk meter MTO tetapi ini adalah di luar lingkungan projek ini.

Aturcara tersebut mesti menerima butiran seperti berikut:

1. tarikh-mula,
2. tarikh-tamat.

Laporan Penyelarasan Mesin Jualan tersebut mesti mengandungi maklumat seperti berikut:

(a) jumlah nilai dipindah dari akaun pelanggan ke antaramuka mesin dari tarikh-mula hingga dan termasuk tarikh-tamat;
(b) nilai pemindahan dari akaun pelanggan ke antaramuka mesin yang direkod pada tarikh-mula dalam akaun pelanggan dan hari sebelum tarikh-mula pada meter MTI mesin jualan;
(c) nilai pemindahan dari akaun pelanggan ke antaramuka mesin yang direkod pada hari selepas tarikh-tamat dalam akaun pelanggan, tetapi bukan pada tarikh-tamat pada meter MTI mesin jualan;
(d) nilai pemindahan balik dari mesin jualan ke akaun pelanggan yang direkodkan pada meter MTI, tetapi tidak terdapat pada mana-mana akaun pelanggan;
(e) jumlah bagi (a) – (b) + (c) + (d);
(f) jumlah nilai bagi amaun yang direkod pada meter MTI;
(g) jika (e) ≠ (f), tajuk ‘Meter mismatch’ dan amaun bagi (e) – (f).

Laporan Penyelarasan Mesin Jualan tersebut harus disusun atur seperti dalam contoh di Figure 5.

```
VENDING MACHINE RECONCILIATION REPORT
FOR DD-MM-YY TO DD-MM-YY

Account-to-machine transfers
     As shown in customer accounts              10,872.50
     Less transactions occurring in the              5.20
     previous machine day
     Plus transactions logged in the next            8.50
     customer day that occurred in the
     current machine day
     Plus reversed transfers                        14.00
                                                ---------
     Sub-total                                  10,889.80
     Vending machine metered value              10,889.60
                                                ---------
     Meter mismatch                                  0.20
```

**Figure 5 – Contoh Laporan Penyelarasan Mesin Jualan**

Aturcara tersebut mesti mengekstrak maklumat yang diperlukan untuk menghasilkan Laporan Penyelarasan Mesin Jualan dari pangkalan data syarikat tersebut. Pangkalan data tersebut mengandungi jadual-jadual berikut yang berkaitan dengan laporan tersebut:

```
create table custTx (
     custId         integer not null,
     custDate       date not null,
     txNo           integer not null
);
create table vendTx (
     machId         integer not null,
     machDate       date not null,
     txNo           integer not null
);
create table txData (
     txNo           integer not null,
     txTime         timestamp not null,
     txType         smallint not null,
     txValue        double precision not null
);
create table vendMeters (
     machId         integer not null,
     machDate       date not null,
     meterId        smallint not null,
     meteredValue   double precision not null
);
```

`custTx`
: ialah satu jadual yang mengandungi satu baris bagi setiap urusniaga yang terdapat pada akaun pelanggan.

`custTx.custId`
: ialah nombor identiti pelanggan.

`custTx.custDate`
: ialah tarikh bagi urusniaga dalam penyata pelanggan.

`custTx.txNo`
: ialah nombor urusniaga yang dapat menentukan urusniaga tersebut secara unik. `custTx` dan `txData` boleh disatukan melalui `txNo` untuk memperoleh maklumat tambahan berkenaan dengan urusniaga tersebut.

`vendTx`
: ialah satu jadual yang mengandungi satu baris bagi setiap urusniaga yang mempengaruhi meter-meter pada mesin jualan.

`vendTx.machId`
: ialah satu nombor identiti mesin yang dapat menentukan sesebuah mesin jualan secara unik.

`vendTx.machDate`
: ialah tarikh urusniaga dalam akaun mesin, yang di mana akan dihantar ke depan pada kira-kira jam 6 pagi.

`vendTx.txNo`
: ialah satu nombor urusniaga yang dapat menentukan urusniaga tersebut secara unik. `vendTx` dan `txData` boleh disatukan melalui `txNo` untuk memperoleh maklumat tambahan berkenaan dengan urusniaga tersebut.

`txData`
: ialah satu jadual yang mengandungi satu baris bagi setiap urusniaga yang mempengaruhi sama ada akaun pelanggan atau akaun mesin.

`txData.txNo`
: ialah nombor urusniaga yang dapat menentukan urusniaga tersebut secara unik.

`txData.txTime`
: ialah masa nyata bagi urusniaga.

`txData.txType`
: ialah satu kod jenis urusniaga. Kod-kod jenis urusniaga dipaparkan di Table 2.

`txData.txValue`
: ialah nilai urusniaga dalam sen. Contohnya, $12.34 akan direkod sebagai 1234.0.

`vendMeters`
: ialah satu jadual yang mengandungi satu baris bagi setiap meter untuk setiap hari mesin.

`vendMeters.machId`
: ialah satu nombor identiti mesin yang dapat menentukan mesin jualan secara unik.

`vendMeters.machDate`
ialah tarikh bagi akaun mesin.

`vendMeters.meterId`
ialah satu nombor identiti meter yang dapat menentukan meter tersebut secara unik. Nilai nombor identiti meter tersebut dipaparkan dalam Table 3.

`vendMeters.meteredValue`
ialah perbezaan bacaan meter dalam sen di antara penutupan dan pembukaan selepas mengambil kira isu-isu seperti bacaan terlebih had meter.

**Table 2 – Kod-kod jenis urusniaga**

| Kod | Erti |
|---|---|
| 1 | Pindahan dari akaun pelanggan ke antaramuka mesin |
| 2 | Pindahan dari antaramuka mesin ke akaun pelanggan |
| 3 | Kemasukkan wang kertas ke dalam mesin jualan |
| 4 | Jualan |
| 5 | Cubaan untuk memindah dari antaramuka mesin ke komputer pusat yang tidak berjaya dan diterbalikkan oleh antaramuka mesin dengan menghantar wang tersebut kembali ke meter kredit pada mesin jualan. |

**Table 3 – ID Meter**

| Kod | Erti |
|---|---|
| 0 | Wang yang dipindah masuk dari antaramuka mesin (MTI) |
| 1 | Wang yang dipindah keluar ke antaramuka mesin (MTO) |
| 2 | Wang kertas yang dimasukkan oleh pelanggan (BI) |
| 3 | Jualan bagi setiap barang (S) |

Sistem syarikat tersebut menambah baris ke dalam pangkalan data seperti yang ditunjukkan dalam Table 4.

**Table 4 – Jadual-jadual yang dikemaskini oleh urusniaga**

| Urusniaga | Jadual-jadual di mana baris-baris (rekod) ditambah |
|---|---|
| Pindahan dari akaun pelanggan ke antaramuka mesin | `custTx, vendTx, txData` |
| Pindahan dari antaramuka mesin ke akaun pelanggan | `custTx, vendTx, txData` |
| Kemasukkan wang kertas ke dalam mesin jualan | `custTx, vendTx, txData` |
| Jualan | `custTx, vendTx, txData` |
| Cubaan untuk memindah dari antaramuka mesin ke komputer pusat yang tidak berjaya dan diterbalikkan oleh antaramuka mesin dengan menghantar wang tersebut kembali ke meter kredit pada mesin jualan | `vendTx, txData` |

| | |
|---|---|
| Mengutip wang kertas dari pengesah wang kertas pada penamatan setiap hari mesin kira-kira 6am. | `vendMeters` |

Pengaturcara yang mengaturcara dalam C atau C++ boleh menggunakan fungsi-fungsi penukaran tarikh seperti ditunjuk dalam Figure 6. Pengaturcara-pengaturcara Java dan C# harus menggunakan fungsi-fungsi penukaran tarikh yang dibekalkan oleh perpustakaan bahasa Java dan C#.

```
#include "dtime.h"
                            // Include declarations of double time
                            // functions.
typedef double Dtime_t;
                            // Double time data type.  Dates and times are
                            // stored in seconds since 00:00 on
                            // 1 January 1970.  Dates are stored as seconds
                            // since 00:00 on 1 January 1970 until 00:00 on
                            // the stored date.
static const Dtime_t NULL_DTIME = -(65536.0*65536.0*65536.0*65536.0);
                            // Reserved double time value (before the big
                            // bang) used to represent an invalid time.
Dtime_t PackDtime (int yr, int mo, int dy, int hr, int mi, int se);
                            // Pack a double time value from its
                            // components.  yr is a 4-digit year (e.g.
                            // 2008).  mo is the calendar month number
                            // (e.g. 1 for January).  dy is the day of the
                            // month (e.g. 1 for the first of the month).
                            // hr, mi and se are the hour, minute and
                            // second components of a 24-hour clock.
                            // PackDtime returns NULL_DTIME if the
                            // components do not represent a valid date
                            // and time.
void UnpackDtime (Dtime_t dtime, int *yr, int *mo, int *dy,
       int *hr, int *mi, int *se);
                            // Unpack a double time value into its
                            // components.
Dtime_t DtimeFromSql (const char *sqlTime);
                            // Convert an SQL date or timestamp string into
                            // double time format.  DtimeFromSql returns
                            // NULL_DTIME if the SQL date or timestamp
                            // string is invalid.
char *SqlDateFromDtime (char *sqlDate, Dtime_t dtime);
                            // Load an SQL date string from a double time
                            // value.  SqlDateFromDtime returns sqlDate.
char *SqlTimestampFromDtime (char *sqlTimestamp, Dtime_t dtime);
                            // Load an SQL timestamp string from a double
                            // time value.  SqlTimestampFromDtime returns
                            // sqlTimestamp.
Dtime_t DtimeFromUserDate (const char *userDate);
                            // Convert a user date in DD-MM-YY format into
                            // the double time format.
char *UserDateFromDtime (char *userDate, Dtime_t dtime);
                            // Convert the date components of a double time
                            // value into a user date string in DD-MM-YY
                            // format. UserDateFromDtime returns userDate.
char *UserTimeFromDtime (char *userTime, Dtime_t dtime);
                            // Convert the time components of a double time
                            // value into a user time string in HH:MM:SS
                            // format. UserTimeFromDtime returns userTime.
```

**Figure 6 – Fungsi-fungsi Masa dan Tarikh**

# 4 PENJANA DATA

Tulis satu aturcara penjana data untuk pangkalan data yang diterangkan dalam Soalan 3.

Penjana data tersebut mesti menjana data bagi tempoh 1 Januari 2010 (tarikh-mula) sehingga 30 Jun 2010 (tarikh-tamat).

Penjana data tersebut mesti beroperasi dalam satu domain masa maya yang bermula pada jam 00:00 untuk tarikh-mula dan maju sehingga mencapai jam 24:00 untuk tarikh-tamat. Penjana data tersebut mesti mengenalpasti perkara yang perlu dilakukan seterusnya dan kemudian memajukan masa maya sehingga masa tersebut, diikuti dengan melakukan perkara tersebut, dan kemudian sekali lagi mengenalpasti perkara yang perlu dilakukan seterusnya.

Penjana data tersebut mesti melakukan emulasi kelakuan bagi 100 pelanggan. Emulator bagi setiap pelanggan mesti melakukan perkara seperti berikut serentak dengan emulator bagi pelanggan-pelanggan yang lain.

1. Menetapkan baki permulaan bagi pelanggan di antara $10 hingga $100 secara pseudo rawak.
2. Ulang berikut apabila masa maya adalah lebih awal dari 24:00 bagi tarikh-tamat:
    a. Pilih masa urusniaga yang seterusnya di antara masa maya sekarang sehingga 48 jam yang akan datang dengan tempoh 8am hingga 8pm mempunyai kebarangkalian 5 kali lebih tinggi daripada kebarangkalian bagi tempoh yang lain.
    b. Tunggu sehingga masa maya mencapai masa urusniaga yang seterusnya.
    c. Pilih satu mesin jualan daripada 20 alternatif secara pseudo rawak.
    d. Jika mesin jualan tersebut sedang diguna:
        i. Tunggu secara giliran sehingga mesin tersebut sedia diguna.
    e. Masukkan kad ke dalam mesin jualan.
    f. Tunggu masa maya berlalu untuk satu tempoh pseudo rawak di antara 5 hingga 15 saat.
    g. Pilih nilai belian di antara $2 hingga $20.
    h. Jika baki pelanggan tidak cukup untuk membayar belian tersebut:
        i. Pilih satu nilai masukan wang kertas di antara baki minimum yang diperlukan untuk melengkapkan belian tersebut sehingga $100.
        ii. Laksanakan emulasi di mana pelanggan memasukkan wang kertas ke dalam mesin jualan dengan nilai wang kertas yang dipilih.
        iii. Tunggu masa maya berlalu untuk satu tempoh pseudo rawak di antara 5 hingga 15 saat.
    i. Laksanakan emulasi jualan di mana nilainya bersamaan dengan nilai belian.
    j. Tunggu masa maya berlalu untuk satu tempoh pseudo rawak di antara 5 hingga 15 saat.
    k. Laksanakan:
        i. Dengan kebarangkalian 1 per 200:
            1. Laksanakan emulasi di mana cubaan untuk menghantar baki pelanggan kembali ke komputer pusat menghadapi kegagalan komunikasi;

ii. Jika tidak:
1. Laksanakan emulasi di mana baki pelanggan berjaya dihantar balik ke komputer pusat;
iii. Tamat jika.
iv. Tunggu masa maya berlalu untuk satu tempoh pseudo rawak di antara 5 hingga 15 saat.

l. Sehingga emulasi di mana penghantaran baki pelanggan kembali ke komputer pusat adalah berjaya.
m. Bebaskan mesin jualan tersebut kepada sebarang proses lain yang menunggunya.

3. Tamat ulang.

Serentak dengan melakukan emulasi tindakan pelanggan, penjana pangkalan data tersebut mesti juga melakukan emulasi tindakan juruteknik mesin jualan. Bagi setiap mesin jualan, satu emulator juruteknik diperlukan. Emulator juruteknik mesin tersebut mesti:

1. Ulang:
    a. Secara pseudo rawak, pilih satu masa hantar-ke-depan yang terletak di antara 05:00 hingga 07:00 hari yang berikutnya.
    b. Jika masa hantar-ke-depan adalah lebih lewat dari atau bersamaan dengan 24:00 bagi tarikh-tamat, hentikan ulangan.
    c. Tunggu masa maya berlalu sehingga mencapai masa hantar-ke-depan.
    d. Jika mesin jualan tersebut sedang diguna:
        i. Tunggu dalam giliran sehingga mesin tersebut sedia diguna.
    e. Laksanakan emulasi pengutipan wang dari mesin jualan dan tambahkan satu baris baru ke vendMeters dalam pangkalan data.
    f. Tunggu masa maya berlalu untuk satu tempoh pseudo rawak di antara 5 hingga 15 saat.
    g. Bebaskan mesin jualan tersebut kepada sebarang proses lain yang menunggunya.
2. Tamat ulang.