E-GENTING
PROGRAMMING COMPETITION 2003

General instructions:

1. Answer one or more of the questions provided.
2. The competition is an open book test.
3. The duration for the competition is 8 hours.
4. Discussion between participants is not allowed.
5. Credit will be given for work-in-progress, especially notes such as data flow diagrams and state transition diagrams or tables.
6. Additional credit will be given to those who can answer more than one question.
7. 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.
8. The marks allocated for the questions are as follows:

No	Name	Marks
1.	Microbank	200
2.	M44	400
3.	Windscreen Wipers	400
4.	25-Factorial	100

9. You should make use of all the standard library functions of the chosen language.
10. You are NOT expected to answer all questions.
11. The questions may be answered in any mainstream programming language.

1 MICROBANK

The government of a struggling third-world economy has established a bank, known as the 'Microbank' to provide relatively small loans to would-be entrepreneurs to assist them to set up new businesses. So far, around a million of these micro-loans have been made. Most of the loans are being serviced (i.e. interest, such as it is, is being collected and/or the capital of the loan is being repaid), but a good proportion of the loans are non-performing (i.e. not being serviced).

Before a loan was approved, two authorisations were required. First, an executive of the Microbank had to recommend a loan be made to a customer. Second, another executive had to approve the granting of the loan.

The Microbank now suspects that some of its executives might have been recommending and approving significant numbers of loans to their friends and relatives without taking any reasonable steps to ensure that the borrowers had either the capacity to service the loans or intention of doing so. The Microbank would like to know the number and value of non-performing loans for each executive, irrespective of whether the executive recommended or approved the loan.

Specifically, it requires a report with the following contents:

1. for each executive:
    a. executive code (6 digits),
    b. executive name (20 characters),
    c. number of loans recommended or approved by the executive,
    d. dollar value of loans recommended or approved by the executive,
    e. number of non-performing loans recommended or approved by the executive,
    f. dollar value of non-performing loans recommended or approved by the executive;
2. total number of loans issued,
3. total value of loans issued,
4. total number of non-performing loans,
5. total value of non-performing loans.

The totals cannot be a simple sum of the corresponding values for each executive because each loan will usually be associated with two executives. If the values for each executive were simply totalled up, they would be approximately twice the required total. They will not be exactly twice because some executives are no longer with the bank and the records for those executives have been deleted.

The data for the report is stored on an SQL database. The relevant parts of the schema are:

```
create table executives (
    execCode  integer not null,
    execName  char(20) not null,
);
```

```
create table loans (
     loanId         integer not null,
     loanCust       char(20) not null,
     loanAmount     double precision not null,
     loanStatus     smallint not null
);
create table approvals (
     appLoanId      integer not null,
     appExecCode    integer not null,
     appType        smallint not null
);
create index appLoanIdInd on approvals(appLoanId);
```

`execCode`
: is the six-digit executive code.

`execName`
: is the executive's name.

`loanId`
: is an 8-digit loan identifier. Each loan has a unique loan identifier.

`loanCust`
: is the customer's name.

`loanAmount`
: is the loan amount in cents. I.e. a $100.00 loan would have `loanAmount`=10000.0.

`loanStatus`
: is 1 if the loan is being serviced, 0 if the loan is not being serviced. Values other than 1 and 0 are invalid.

`appLoanId`
: is the loan identifier to which the approval relates. It is in the same domain as the `loans.loanId` column.

`appExecCode`
: is the executive code of the Microbank executive who recommended or approved the loan. It is in the same domain as the `executives.execCode` column.

`appType`
: is 0 for a recommendation or 1 for an approval. Values other than 0 and 1 are invalid.

It is anticipated that the reporting program will be run on a daily basis to review changes in the status of the loans recommended and approved by the executives. Consequently, the execution time of the reporting program must be less than two hours.

Various database access operations have been timed in prior research. A simple, unsorted row-fetch from a cursor takes, on average, 0.8ms. A select operation that can make use of the index takes, on average, 1.5ms. Select operations that cannot make use

of an index take approximately 0.5n ms to complete, where ‘n’ is the number of rows in the table.

There are approximately 1,000 executives and 1,000,000 loans on the database.

The tables are only a small part of a much larger database. In order to create a new index, the entire database must be unloaded onto tape, the index can then be created and then the contents of the tape must be reloaded. This process takes several weeks. The database is also used for mission-critical real-time transaction processing that cannot be suspended for the several weeks it would take to unload and reload the database. In a practical sense, it is impossible to add new indexes.

Your task is to write the reporting program so that it satisfies all the above requirements.

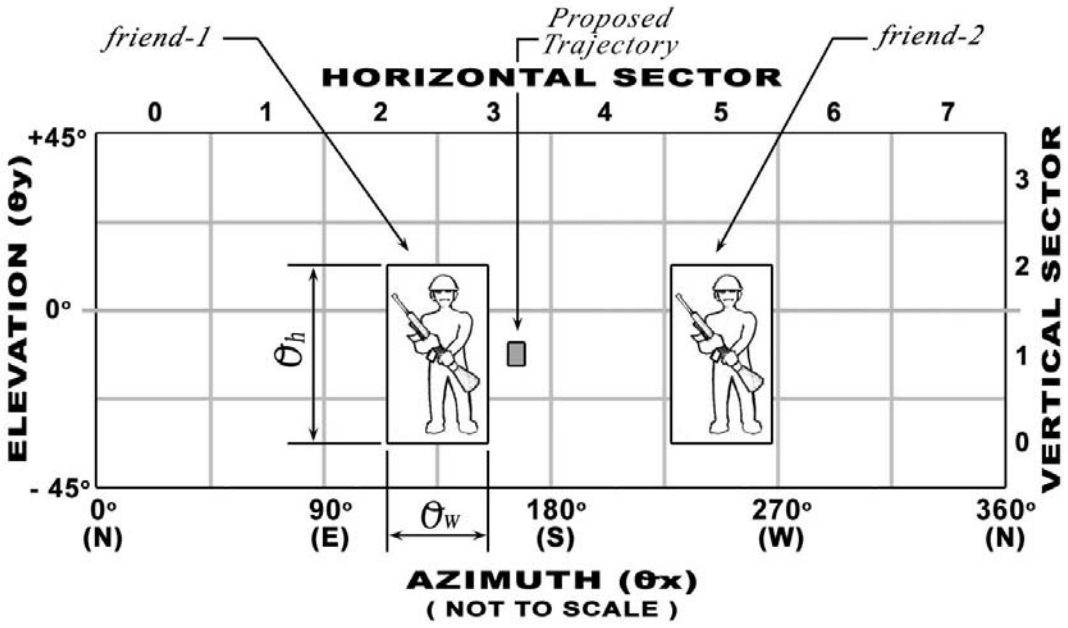
2 M44

In a recent conflict, the soldiers of Nanopia, a technologically advanced first-world democracy, turned out to be much better at killing each other than the enemy was at killing them. As a consequence of that unfortunate statistic, the Congress of Nanopia passed a bill that restricted its soldiers to a new type of rifle known as the M44. The M44 incorporates a transponder that emits a signal that is received by all the other M44s within a 25km radius. When the trigger of the M44 is pulled, the weapon senses its orientation and checks whether the bullet is likely to hit any of the holders of the other M44s in the vicinity. The firing mechanism is not released until this check is satisfactorily completed.

The soldiers of Nanopia are now engaged in an urban peacekeeping operation. Nanopia has between 60,000 and 100,000 soldiers spread among the population of an urban centre approximately 50km in diameter. This has resulted in a large number of potential ‘friends’ being stored in each M44. The M44s are not firing a round until 1 to 3 seconds after the trigger is pulled. This is because of the large number of potential friends that the M44s must check before releasing the firing mechanism. The urban guerrillas that the Nanopian army was supposed to bring under control are taking advantage of the delay and the Nanopian casualties are mounting.

In a near-Newtonian experience with a high-velocity sniper round, a member of the Nanopian Engineers came up with the idea that it might be possible to divide the M44’s field of fire into sectors. The location of each friend could then be recorded in a separate table for each sector. If this were done it would only be necessary to search the sectors that the proposed trajectory intersected in order to determine if the trajectory was likely to hit a friend. This might reduce the search time from 1 or 3 seconds down to an acceptable period of 10 or 20 milliseconds.

Consider the field of fire of the M44 in polar co-ordinates represented by un-projected azimuth and elevation:



Azimuth is the direction the gun is pointing in, in degrees east of North. Elevation is the angle between the barrel of the gun and the horizon. If the gun is horizontal, its elevation is 0º; if it is pointing up in the air at an angle of 30º to the horizontal, its elevation is +30º.

The existing M44 program represents each friend by a rectangle in the azimuth-elevation co-ordinate system. The trajectory of the bullet is also represented by a rectangle. The rectangle of a friend or trajectory is referred to as the 'profile' of the friend or trajectory.

The Nanopian Engineer proposes that a hash-type algorithm be used to dramatically reduce the search time needed to determine whether or not any particular trajectory is likely to hit a friend. He suggests that a hash-bucket be created for each combination of horizontal and vertical sectors.

For the purposes of describing the algorithm, each sector is identified by an expression of the form '(x,y)', where 'x' is the horizontal sector number and 'y' is the vertical sector number. Each hash-bucket is represented by the term H(x,y), where '(x,y)' is the sector associated with the hash-bucket.

The engineer proposes that a reference to each friend be put in the hash bucket of each sector that the profile of the friend intersects. For example, in the above diagram, there would be a reference to friend-1 in hash buckets, H(2,0), H(2,1), H(2,2), H(3,0), H(3,1) and H(3,2) and a reference for friend-2 in hash buckets H(5,0), H(5,1) and H(5,2).

It is possible by reasonably obvious mathematical means to determine the sectors that a proposed trajectory intersects. In the example, the trajectory only intersects sector (3,1), though it is certainly feasible for a trajectory to intersect more than one sector. To determine whether or not a proposed trajectory is likely to hit a friend, it is only

necessary to check the friends with references in the hash-buckets corresponding to the sectors that the trajectory intersects. In the example, it is only necessary to check the friends with references in hash-bucket H(3,1).

Just because a trajectory and a friend both intersect sector (3,1), it is not necessarily the case that the trajectory will hit the friend. It is still necessary to check whether the profiles of the friends referenced in the hash-bucket intersect the profile of the trajectory.

Your task is to create a prototype of the engineer's algorithm to test its performance. If the prototype works satisfactorily, your company will win a multi-million dollar contract to provide field upgrades for the M44s and your stock options will transmogrify into a retirement annuity. You will also save the lives of hundreds of soldiers.

The prototype must construct a rectangular theatre of operations 50km by 50km. The subject M44 will be located in the centre of the rectangle. 100,000 friends must then be pseudo-randomly positioned in a uniform distribution over the rectangular theatre, but no closer to the subject than 5m and no further from the subject than 25km. The elevation of each friend is to be pseudo-randomly generated in a uniform distribution, with the centre of each friend having an elevation of between -30º and +30º with respect to the subject. The dimensions of the profile of each friend can then be calculated by assuming that each friend is 1.8m high and 0.5m wide.

The prototype must transform the dimensions of each friend into polar co-ordinates and load the hash buckets. Care must be taken in dealing with friends to the north of the subject because such friends may appear in both high-numbered and low-numbered sectors.

The prototype must then record a starting time and generate 1,000 pseudo-randomly selected firing trajectories. The trajectories may have any azimuth between 0º and 360º, but the elevations are to be restricted to -40º to +40º. The profile of each trajectory may be taken to be 0.2º wide by 1º high. The prototype must determine whether or not each trajectory intersects a friend and store the profile of the trajectory and the identities of the friends, if any, that it intersects. After performing the 1,000 bucket-based intersection determinations, the prototype must record the ending time and then list the profile of each trajectory and those of the friends, if any, that it intersects. The listing must contain the polar co-ordinates (i.e. θx and θy) of the centre of each profile and its width (θw) and height (θh) in degrees. It must then display the average computation time per trajectory.

The prototype must use symbolic constants to define the horizontal and vertical sector densities. If the sector size is too small, the profile of the trajectory will intersect a large number of sectors and thereby increase the search time. If the sector size is too big each sector will contain a large number of friends and the efficiency of the algorithm will, once again, be compromised. It will probably be necessary to adjust the horizontal and vertical sector densities to determine optimum values.

The Nanopian Engineer presented the following pseudo-code of the prototype to the Joint Chiefs to convince them to provide funding for the project. You may use it as a base for developing the prototype.

1. Initialise the bucket data structures.

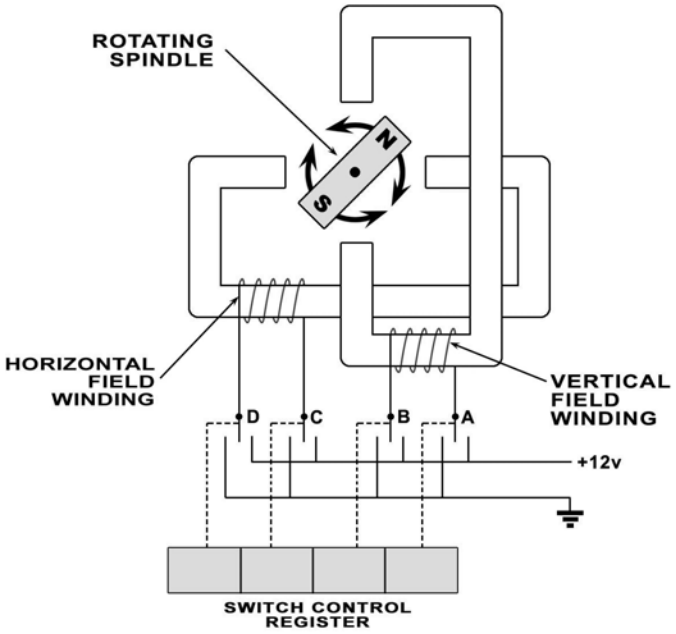
2. Repeat the following until the profiles of 100,000 friends have been loaded into the bucket data structures:
    a. Generate a pseudo-random easterly component (e) of the distance from the subject to the friend in the range –25 to +25km.
    b. Generate a pseudo-random northerly component (n) of the of the distance from the subject to the friend in the range –25 to +25km.
    c. Calculate the horizontal distance to the friend (r) using the formula $r = \text{sqrt}(e^2 + n^2)$.
    d. If the horizontal distance to the friend is outside the range 5m to 25km, discard the location and generate another pair of co-ordinates.
    e. Calculate the azimuth of the centre of the friend (θx), in principle by the use of the formula θx = atan(e / n), but adjusting for the correct quadrant when n is close to zero and/or when one or both of the location components are negative and, if necessary, converting radians into degrees.
    f. Generate the pseudo-random elevation of the friend (θy) in the range –30º to +30º.
    g. Calculate the angular width (θw) and height (θh) of the friend using the following formulas:
        $\theta w = (W * 180) / (r * \pi)$
        $\theta h = (H * 180) / (r * \pi)$
        Where W is the physical width of the friend (assumed to be 0.5m) and H is the physical height of the friend (assumed to be 1.8m).
    h. Load the profile of the friend represented by θx, θy, θh and θw into the bucket data structures.
3. Record the starting time (t_s).
4. Repeat 1,000 times:
    a. Generate a pseudo-random firing azimuth in the range 0º to 360º.
    b. Generate a pseudo-random firing elevation in the range –40º to +40º.
    c. Determine the profile of the trajectory given that it is 0.2º wide by 1º high.
    d. Store the profile of the trajectory.
    e. Use the bucket data structure to determine which, if any, of the friends the proposed firing direction is likely to hit.
    f. For each friend the proposed trajectory is likely to hit:
        i. Store the identity of the friend.
5. Record the ending time (t_e).
6. For each generated trajectory:
    a. Display the profile of the trajectory.
    b. For each friend the trajectory is likely to hit:
        i. Display the profile of the friend.
7. Calculate and display the average time per trajectory $(t_e - t_s)/1000$.

Although the profile of the friend is generated in terms of θx, θy, θh and θw, these may not be the most convenient values to use for the purposes of determining profile intersections. The prototype is not required to store θx, θy, θh and θw. It may store the profile in a form more suited to determining profile intersections.

3 WINDSCREEN WIPERS

In an effort to reduce the number of mechanical parts in a windscreen wiper mechanism, an automobile manufacturer intends to replace its current DC motor mechanism with a brush-less stepping motor mechanism.

Conceptually, the stepping motor consists of a permanent magnet mounted on a spindle with two field windings that can be energised in sequence to drive the spindle in one direction or the other. The diagram below represents the mechanism of the stepping motor:



When switch A is switched to +12V and switch B is switched to ground (i.e. 0V), the conventional current will flow from the common of switch A to the common of switch B. This will cause the core of the A-B winding to be magnetised so that its top is a north pole and its bottom is a south pole. This, in turn will cause the spindle to rotate clockwise until the north pole of the spindle is at the bottom. By operating switches A to D in the correct order, the spindle can be made to rotate in either direction.

Switches A to D are controlled via a 4-bit switch control register (SCR) that can be set by software. The least significant bit of the SCR operates switch A and the most significant bit operates switch D. When a SCR bit is '0', the common of the switch is connected to ground; when the bit is '1', the common of the switch is connected to +12V.

The SCR can be set from C or C++ by calling the following function:

```
void SetSCR (int scrVal)
```

The SCR can be set from Java by calling the following static method:

```
class WiperControl {
     //...
     public void setSCR (int scrVal)
     //...
}
```

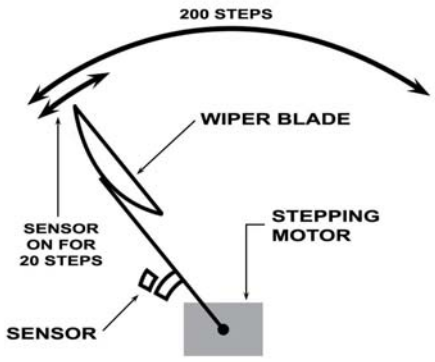
In all of C, C++ and Java, `scrVal` is the value to be loaded into the SCR. Only the four least significant bits of `scrVal` are loaded into the SCR. The higher order bits are ignored.

To turn the spindle, the windings must be energised in either the ascending or descending sequence of the rows in the following table. If the windings are energised in the descending sequence, the motor will rotate clockwise. If the windings are energised in the ascending sequence, the motor will rotate anticlockwise.

For clockwise rotation	**SCR VALUE**				**For anti-clockwise rotation**
	D	**C**	**B**	**A**	
↓	0	0	0	1	↑
	0	1	0	0	
	0	0	1	0	
	1	0	0	0	

Each row in the above table is referred to as a 'step'. In the windscreen wiper mechanism, the stepping motor is geared down so that there are 200 steps from the left-most position of the wiper blade to the right-most position of the wiper blade. The gearing mechanism does not reverse the direction of the stepping motor. If the stepping motor is rotating clockwise, then the blades will move from left to right and vice versa.

There is a sensor that detects the position of the wiper blades when they are close to the left-most limit of their travel. The following diagram shows how the sensor works.



As the wiper blade swings from the right to the left, the sensor will switch from its off state to its on state. The wiper blade must stop and reverse its direction 20 steps after this transition.

The state of the sensor can be sensed from C or C++ by calling the following function:

```
int GetSensorState()
```

return value
: is zero if the sensor is off, or non-zero if the sensor is on.

It can be sensed in Java by calling:

```
class WiperControl {
     //...
     public boolean getSensorState()
     //...
}
```

return value
: is false if the sensor is off, or true if the sensor is on.

The normal operating speed of the motor is 200 steps per second, but if it is switched from stopped directly to 200 steps per second it will almost certainly skip a few steps and the accelerations involved, in time, will damage the reduction gear and wiper blade mechanism.

The maximum acceptable acceleration is 1,250 steps per second squared in either direction. This acceleration is achieved, within the limits required by the mechanism, by using the sequence of inter-step times listed in the table below when starting, stopping and reversing the direction of the wiper blade.

Inter-Step Time (milliseconds)															
42	17	13	11	10	9	8	8	7	7	6	6	6	6	6	5

Your task is to write a computer program to control the windscreen wiper stepping motor. The program may be written in C, C++ or Java.

If you write in C or C++, your program must contain the following functions that will be called by the operating system of the microprocessor.

```
void Initialise ();
void StartWiping ();
void StopWiping ();
void Tick ();
```

If you write in Java, your program must implement the following interface.

```
interface WiperInterface {
     void initialise (WiperControl control);
     void startWiping ();
     void stopWiping ();
     void tick ();
}
```

In all the languages, the `Initialise` function must first switch off all the field windings by setting the SCR to zero. It must then initialise any internal data structures and then test the state of the sensor. If the sensor is off, it must initiate a process to return

the wiper blades to the left-most starting position at the maximum non-progressive speed of 42ms per step. When the blades are in the left-most starting position, the control program must switch off the field windings by setting the SCR to zero. If the sensor is on when `Initialise` is called, the wiper blades must stay in their initial position.

The Java version of the `initialise` method receives a reference to an instance of the `WiperControl` class that can be used to read the state of the sensor and set the SCR.

The `StartWiping` function must start the wiper blades moving back and forth. The `StartWiping` function may be called at any time after `Initialise` is called. If the `StartWiping` function is called while the blades are being returned to the left most starting position at 42ms per step, the return process must be completed before the normal back and forth cycle is initiated. Once the wiper blades are in the left most starting position, the program must accelerate the blades to their maximum speed in the left-to-right direction in accordance with the acceleration table. The program must then decelerate the blades so that they stop 180 steps after the sensor switches off. It must then reverse the direction of the motor, accelerate the blades, adjust the theoretical location of the blades when the sensor switches on, decelerate and reverse the direction of the motor and so on until the `StopWiping` function is called.

The `StopWiping` function may be called at any time after `Initialise` is called, even when the blades are already stationary. In all cases, the wiper blades must be allowed to return to the left-most starting position in the normal course of the cycle that was in progress when `StopWiping` was called. The motor windings must then be switched off by setting the SCR to zero.

The `Tick` function is called by the operating system once every millisecond.

The control program must not lock control in any of the application functions. If control is locked in a loop in the `StartWiping` function, the `StopWiping` function will never be called.

4 25-FACTORIAL

Write a program to calculate and display the exact decimal value of 25-factorial (i.e. 25 * 24 * 23 * ... * 3 * 2) without the aid of any language or operating system supplied extended-precision arithmetic functions.

25-factorial is of the order of 2^{84} or 10^{26}. C and C++ have 32-bit long integers and Java has 64-bit long integers, neither of these have sufficient precision to perform the calculation. In all the languages, the maximum precision of a floating-point type is of the order of 52 bits.

You should carefully consider the way in which the extended precision number is to be stored. Some storage formats will make the task quite easy, others, in comparison, will more than double the amount of effort involved.

PERTANDINGAN PENGATUCARAAN E-GENTING 2003

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. 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
8. Markah yang diperuntukkan kepada setiap soalan adalah seperti berikut:

No	Nama	Markah
1.	Mikrobank	200
2.	M44	400
3.	Pengelap Cermin Penahan Angin	400
4.	Faktorial-25	100

9. Anda harus menggunakan fungsi-fungsi perpustakaan piawai bagi bahasa pengaturcaraan yang dipilih.
10. Anda TIDAK dijangka akan menjawab semua soalan.
11. Soalan-soalan boleh dijawab dalam mana-mana satu bahasa pengaturcaraan aliran utama.

1 MIKROBANK

Sebuah kerajaan dunia ketiga yang mengalami kemerosotan ekonomi telah menubuhkan sebuah bank yang dikenali sebagai 'Mikrobank' untuk memberi pinjaman kecil kepada bakal usahawan bagi membantu mereka memulakan perniagaan baru. Setakat ini, lebih kurang sejuta pinjaman-mikro sebegini telah dikeluarkan. Kebanyakan pinjaman ini diserviskan (iaitu faedah pinjaman dipungut dan/atau pokok pinjaman dibayar balik). Tetapi, terdapat sebahagian daripada pinjaman tersebut tidak berbayar (tidak diserviskan).

Sebelum sesuatu pinjaman diberi, dua peringkat kelulusan diperlukan. Pertama, seorang pegawai Mikrobank perlu merekomen supaya pinjaman diberikan kepada seseorang pelanggan. Kemudian, seorang lagi pegawai perlu meluluskan pinjaman tersebut.

Kini, Mikrobank mengesyaki bahawa sesetengah pegawainya merekomen dan meluluskan sebilangan besar pinjaman kepada kawan-kawan dan saudara-mara mereka tanpa mengambil sebarang langkah yang berpatutan untuk menjamin bahawa peminjam-peminjam tersebut sama ada mempunyai kemampuan untuk menjelaskan pinjaman atau niat untuk berbuat demikian. Mikrobank ingin mengetahui bilangan dan nilai pinjaman tidak berbayar bagi setiap pegawai, tanpa mengira sama ada dialah yang merekomen atau meluluskan pinjaman itu.

Khususnya, ia mengehendaki satu laporan dengan isi kandungan yang berikut:

1. Bagi setiap pegawai:
    a. kod pegawai (6 angka),
    b. nama pegawai (20 huruf),
    c. bilangan pinjaman yang direkomen atau diluluskan oleh pegawai itu,
    d. nilai dolar pinjaman yang direkomen atau diluluskan oleh pegawai itu,
    e. bilangan pinjaman tidak berbayar yang direkomen atau diluluskan oleh pegawai itu,
    f. nilai dolar pinjaman tidak berbayar yang direkomen atau diluluskan oleh pegawai itu;
2. jumlah bilangan pinjaman yang dikeluarkan;
3. jumlah nilai pinjaman yang dikeluarkan;
4. jumlah bilangan pinjaman tidak berbayar;
5. jumlah nilai pinjaman tidak berbayar.

Jumlah-jumlah yang tersebut di atas tidak boleh merupakan hasil campuran ringkas nilai-nilai berkenaan bagi setiap pegawai kerana setiap pinjaman sentiasa berkaitan dengan dua orang pegawai. Sekiranya, nilai-nilai bagi setiap pegawai dicampurkan secara ringkas, keputusannya akan menjadi lebih kurang dua kali ganda jumlah yang dikehendaki. Keputusan demikian tidak akan menjadi tepat dua kali ganda kerana sesetengah pegawai tidak lagi bekerja di bank itu dan rekod-rekod mereka telah dihapuskan.

Data untuk laporan ini disimpan di dalam satu pengkalan data SQL. Bahagian-bahagian skema yang berkaitan adalah:-

```
create table exectives (
     execCode        integer not null,
     execName        char(20) not null,
);
create table loans (
     loanId          integer not null,
     loanCust        char(20) not null,
     loanAmount      double precision not null,
     loanStatus      smallint not null
);
creat table approvals (
     appLoanId       integer not null,
     appExecCode     integer not null,
     appType         smallint not null
);
create index appLoanIdInd on approvals (appLoanId);
```

`execCode`
ialah kod pegawai 6-angka.

`execName`
ialah nama pegawai.

`loanId`
ialah identiti pinjaman 8-angka. Setiap pinjaman mempunyai identiti pinjaman yang tunggal.

`loanCust`
ialah nama pelanggan.

`loanAmount`
ialah pinjaman dalam unit sen. Iaitu satu pinjaman bernilai $100.00 akan mempunyai `loanAmount = 10000.0.`

`loanStatus`
ialah 1 jika pinjaman itu diserviskan, 0 jika pinjaman itu tidak diserviskan. Nilai-nilai selain daripada 1 dan 0 adalah tidah sah.

`appLoanId`
ialah identiti pinjaman yang berhubung dengan kelulusan pinjaman itu. Ia berada di dalam domain yang sama dengan lajur `loans.loanId.`

`appExecCode`
ialah kod pegawai Mikrobank yang merekomen atau meluluskan permohonan pinjaman. Ia berada di dalam domain yang sama dengan lajur `executives.execCode.`

`appType`
ialah 0 bagi rekomen atau 1 bagi kelulusan. Nilai-nilai selain daripada 0 dan 1 adalah tidak sah.

Adalah dijangka bahawa program laporan ini akan dilarikan setiap hari untuk mengkaji perubahan ke atas status pinjaman-pinjaman yang direkomen dan diluluskan oleh para pegawai. Oleh yang demikian, masa yang diambil untuk melaksanakan program laporan ini mestilah kurang daripada 2 jam.

Masa yang diambil oleh pelbagai operasi akses pengkalan data telah dicatatkan pada penyelidikan yang terlebih dahulu. Satu ambil-baris secara tidak terisih yang ringkas daripada kursor mengambil, pada puratanya, 0.8 ms. Satu operasi memilih yang boleh menggunakan indeks memakan masa selama 1.5ms pada puratanya. Operasi-operasi yang tidak dapat menggunakan indeks mengambil masa kira-kira 0.5n ms untuk dihabiskan, dimana 'n' merupakan bilangan baris di dalam jadual berkenaan.

Terdapat kira-kira 1,000 pegawai dan 1,000,000 pinjaman di dalam pengkalan data.

Jadual-jadual di atas hanyalah sebahagian kecil daripada pengkalan data yang jauh lebih besar. Untuk membentuk satu indeks yang baru, seluruh pengkalan data mestilah dialihkan ke dalam pita, indeks baru dibentuk selepas ini dan kemudiannya kandungan pita mestilah dialihkan kembali. Proses ini mengambil masa beberapa minggu. Pengkalan data tersebut juga digunakan untuk pemprosesan urusniaga masa nyata misi kritikal yang tidak boleh dihentikan buat sementara bagi tempoh masa selama beberapa minggu yang diambil untuk mengalihkan dan mengembalikan pengkalan data tersebut. Dalam keadaan praktikal, adalah mustahil untuk menambah indeks yang baru.

Tugas anda adalah menulis satu program laporan yang memenuhi semua syarat yang tersebut diatas.

2 M44

Dalam satu konflik baru-baru ini, askar-askar Nanopia, sebuah negara demokrasi

dunia-pertama yang mempunyai teknologi canggih, tanpa disangka, sudah menjadi jauh lebih baik membunuh sesama sendiri berbanding dengan pembunuhan yang dilakukan oleh musuh mereka. Sebagai akibatnya, Kongres Nanopia meluluskan satu rang undang-undang yang menghadkan askar-askarnya kepada sejenis senapang baru yang dikenali sebagai M44. M44 dipasang dengan transponder (Sistem komunikasi yang menerima isyarat dan memancar semula isyarat pada frekuensi yang berbeza) yang memancar satu isyarat yang boleh diterima oleh semua M44 lain dalam lingkungan 25km. Apabila

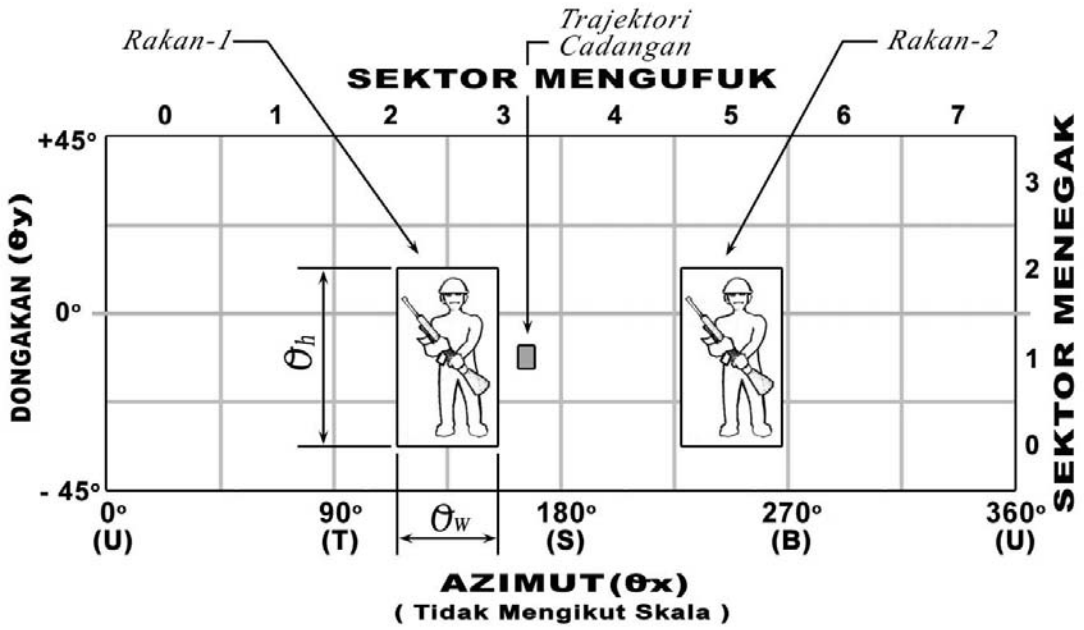
pemicu M44 ditarik, senjata ini akan mengesan orientasinya dan memeriksa sama ada pelurunya berkemungkinan terkena sebarang pemegang M44 lain yang berada di sekelilingnya. Tembakan tidak akan dilepaskan sehingga pemeriksaan siap dilakukan dengan keputusan yang memuaskan.

Sekarang, askar-askar Nanopia terlibat dalam satu operasi pengamanan di sebuah bandar. Nanopia mempunyai kira-kira 60,000 hingga 100,000 askar yang bertaburan di sekeliling pusat bandar ini dan meliputi kawasan yang diameternya lebih kurang 50 km. Ini telah menyebabkan sebilangan besar 'rakan' yang berkemungkinan ditimpa tembakan disimpan di dalam setiap M44. M44 tidak akan melepaskan sesuatu tembakan sehingga 1 sampai 3 saat selepas pemicunya ditarik. Ini disebabkan bilangan besar rakan yang M44 mesti menyemak sebelum melepaskan sebarang tembakan. Kumpulan gerila di dalam

bandar itu yang sepatutnya dibawa kepada keadaan terkawal oleh tentera Nanopia mengambil kesempatan penangguhan tembakan ini dan pengorbanan askar-askar Nanopia meningkat.

Dalam satu pengalaman yang menyerupai Newton dengan peluru yang laju dan tepat, seorang anggota daripada Jurutera Nanopia mengemukakan pendapatnya bahawa medan tembakan M44 mungkin boleh dibahagikan kepada beberapa sektor. Dengan demikian, lokasi setiap rakan yang bertugas boleh dicatatkan dalam jadual yang berasingan bagi setiap sektor yang diliputi. Sekiranya ini dilakukan, M44 hanya perlu mencari sektor-sektor yang bersilang dengan trajektori cadangannya untuk menentukan sama ada trajektori tersebut akan terkena seseorang rakan. Ini mungkin dapat mengurangkan masa cari dari 1 atau 3 saat kepada satu tempoh masa yang boleh diterima seperti 10 atau 20 milisaat.

Cuba fikirkan padang tembakan bagi M44 dalam koordinat berkutub (*polar coordinates*) yang diwakili oleh azimut dan dongakan:



Azimut adalah arah senapang itu dihalakan, dalam darjah dari arah timur laut. Dongakan adalah sudut diantara laras senapang dengan paras ufuk. Jika senapang itu mendatar, dongakannya adalah 0^o; jika ia menghala ke udara pada sudut 30^o dari paras mengufuk, dongakannya ialah $+30^o$.

Aturcara M44 yang wujud pada masa sekarang mewakili setiap rakan dengan segiempat tepat dalam sistem koordinat azimut-dongakan. Trajektori peluru juga diwakili dengan sebuah segiempat tepat. Segiempat tepat yang mewakili rakan atau trajektori peluru adalah dirujuk sebagai *profile* bagi rakan atau trajektori itu.

Jurutera Nanopia itu mengesyorkan supaya satu algoritma jenis-cincangan (*hash-type*) digunakan untuk mengurangkan masa cari yang perlu diambil bagi menentukan sama ada

sesuatu trajektori ada kemungkinannya terkena seorang rakan. Dia mencadangkan agar satu *hash-bucket* digunakan bagi setiap kombinasi sektor mengufuk dan sektor menegak.

Bagi tujuan penghuraian algoritma tersebut, setiap sektor dikenali dengan satu ungkapan dalam bentuk '(x, y)', dimana 'x' ialah nombor sektor mengufuk dan 'y' ialah nombor sektor menegak. Setiap *hash-bucket* diwakili oleh H(x, y), dimana '(x, y)' ialah sektor yang berhubungan dengan *hash-bucket* tersebut.

Jurutera itu mencadangkan supaya satu rujukan kepada setiap rakan ditaruhkan ke dalam *hash-bucket* bagi setiap sektor yang bersilang dengan *profile* rakan tersebut. Sebagai contoh, dalam gambar rajah diatas, terdapat satu rujukan kepada rakan-1 di dalam *hash buckets* H(2,0), H(2,1), H(2,2) H(3,0), H(3,1) dan H(3,2) dan satu rujukan kepada rakan-2 dalam hash buckets H(5,0), H(5,1) dan H(5,2).

Dengan cara matematik yang agak mudah, kita dapat menentukan sektor dimana sesuatu trajektori peluru bersilang. Bagi contoh di atas, trajektori itu hanya bersilang dengan sektor (3,1), walaupun ia memang boleh bersilang dengan lebih daripada satu sektor. Untuk menentukan sama ada sesuatu trajektori cadangan berkemungkinan terkena seorang rakan, kita hanya perlu memeriksa sama ada terdapat rakan dengan rujukan di dalam *hash-buckets* yang sepadan dengan sektor-sektor persilangan trajektori peluru itu. Dalam contoh ini, kita hanya perlu menyemak rakan-rakan yang mempunyai rujukan di dalam *hash-bucket* H(3,1).

Hanya kerana satu trajektori peluru dan seorang rakan masing-masing bersilang dengan sektor (3,1) tidak semestinya menjadikan satu kes dimana trajektori peluru itu akan terkena rakan tersebut. Adalah masih perlu untuk menyemak sama ada *profile* rakan yang dirujuk oleh *hash-bucket* itu bertindih dengan *profile* trajektori peluru itu.

Tugas anda ialah membina satu prototaip bagi algoritma jurutera itu untuk menguji prestasinya. Jika prototaip itu memuaskan, syarikat anda akan memenangi satu kontrak berjuta-juta dolar untuk membekalkan peningkatan mutu bagi M44 dan opsyen saham anda akan bertukar menjadi anuiti persaraan. Anda juga akan menyelamatkan nyawa beratus-ratus askar.

Prototaip itu mestilah membina satu dewan operasi segiempat tepat berukuran 50km x 50 km. Subjek M44 akan ditempatkan di pusat segiempat tepat itu. 100,000 orang rakan kemudian ditempatkan secara pseudorawak mengikut taburan seragam atas dewan segiempat tepat itu, tetapi jarak mereka dari pusat dewan itu mestilah sekurang-kurangnya 5m serta tidak melebihi 25km. Dongakan bagi setiap rakan akan dijana secara pseudorawak mengikut taburan seragam, dengan pusat setiap rakan mempunyai dongakan terhadap subjek M44 di antara -30^{o} dan $+30^{o}$. Dimensi *profile* bagi setiap rakan boleh dikira kemudian dengan menganggap bahawa ketinggian dan lebar setiap rakan adalah 1.8m dan 0.5m masing-masing.

Prototaip itu mestilah mengubah dimensi setiap rakan kepada koordinat berkutub dan memuatkan kesemua *hash bucket*s dengan rujukan-rujukan kepada rakan yang berkaitan. Perhatian mesti diambil apabila menguruskan rakan-rakan yang berada di utara subjek itu kerana rakan-rakan sebegini mungkin wujud di kedua-dua sektor bernombor tinggi dan sektor bernombor rendah.

Prototaip itu kemudian mesti merekod masa permulaan dan menjana 1,000 trajektori tembakan secara pseudorawak. Trajektori-trajektori tersebut boleh mempunyai azimut di antara 0^o dan 360^o, tetapi dongakan mereka dihadkan kepada julat di antara -40^o dan $+40^o$. *Profile* setiap trajektori boleh dianggap mempunyai lebar 0.2^o dan ketinggian 1^o. Prototaip itu mesti menentukan sama ada setiap trajektori bersilang dengan seseorang rakan, menyimpan '*profile*' trajektori itu dan identiti rakan-rakan yang tersilang, jika ada. Selepas melakukan 1,000 penentuan persilangan berdasarkan *bucket*, prototaip itu mesti merekod masa tamat dan seterusnya menyenaraikan *profile* setiap trajektori dan *profile* rakan-rakan yang tersilang, jika ada. Senarai itu mesti mengandungi koordinat berkutub (iaitu θx dan θy) bagi pusat setiap *profile*, lebar (θw) dan tingginya (θh) dalam darjah. Ia kemudian mesti memaparkan masa purata pengiraan komputer untuk satu trajektori.

Prototaip itu mesti menggunakan pemalar bersimbol untuk mentakrifkan ketumpatan sektor mengufuk dan sektor menegak. Jika saiz sektor itu adalah terlampau kecil, *profile* trajektori peluru akan bersilang dengan sebilangan besar sektor dan dengan itu meningkatkkan masa cari. Jika saiz sektor itu terlalu besar, setiap sektor akan mengandungi sekumpulan besar rakan dan kecekapan algoritma itu akan sekali lagi terjejas. Kita mungkin perlu melaraskan ketumpatan sektor mengufuk dan sektor menegak untuk menentukan nilai optimum masing-masing.

Jurutera Nanopia itu mempersembahkan pseudokod berikut bagi prototaip yang dikehendaki kepada ketua-ketua pegawai tentera bersama untuk menyakinkan mereka supaya membiayai projek itu. Anda boleh mnggunakannya sebagai asas pembangunan prototaip itu.

1. Tetapkan struktur-struktur data *bucket* dengan nilai-nilai permulaan.
2. Ulangi yang berikut sehingga '*profile*s' bagi 100,000 rakan sudah dimuatkan ke dalam struktur-struktur data *bucket*:
    a. Jana satu nombor pseudorawak yang mewakili komponen timuran (e) bagi jarak dari M44 kepada rakan di antara –25km dan +25 km.
    b. Jana satu nombor pseudorawak yang mewakili komponen utaraan (n) bagi jarak dari M44 kepada rakan di antara – 25km dan +25 km.
    c. Kira jarak mengufuk dari M44 kepada rakan (r) dengan menggunakan rumusan berikut:
        $r = \text{sqrt}(e^2 + n^2)$.
    d. Jika jarak mengufuk dari M44 kepada rakan berada di luar julat 5m ke 25 km, campakkan lokasi itu dan jana sepasang koordinat yang lain.
    e. Kira azimut pusat rakan (θx) pada prinsip dengan menggunakan rumusan θx = atan(e / n), tetapi membuat pelarasan pembetulan sukuan apabila n berhampiran dengan sifar dan/atau apabila satu atau kedua-dua komponen lokasi adalah negatif. dan, Jika perlu, tukar radians kepada darjah.
    f. Jana sudut dongakan rakan (θy) pseudorawak di antara -30^o dan $+30^o$.
    g. Kira lebar (θw) dan ketinggian (θh) rakan diukur secara sudut dengan menggunakan rumusan berikut:
        $\theta w = (W * 180) / (r * \pi)$
        $\theta h = (H * 180)/ (r * \pi)$
        dimana W merupakan lebar fizikal rakan (dianggap sebagai 0.5m) dan H ialah ketinggian fizikal rakan (dianggap sebagai 1.8m).

h. Muatkan *profile* rakan yang diwakili oleh θx, θy, θh dan θw ke dalam struktur-struktur data *bucket*.

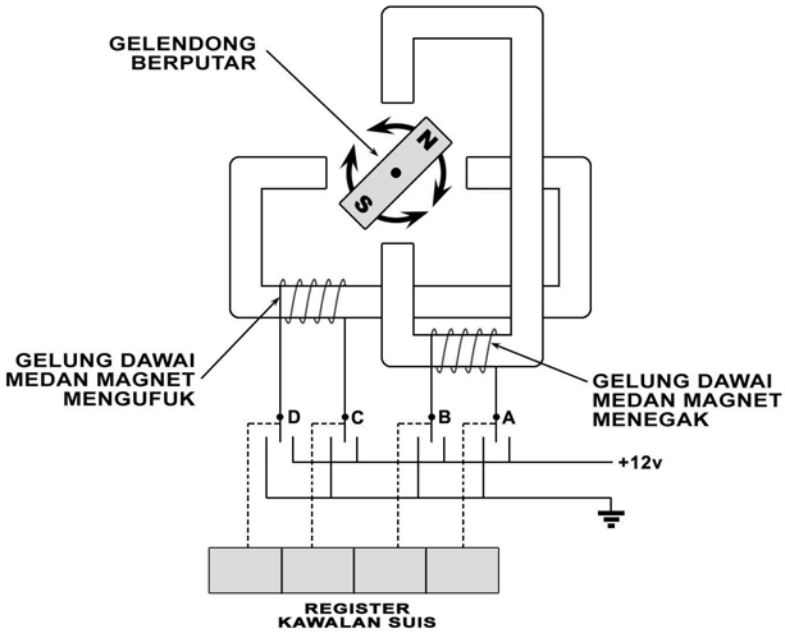
3. Rekod masa permulaan (t_s)
4. Ulangi 1,000 kali:
    a. Jana azimut penembakan pseudorawak di antara 0^o dan 360^o.
    b. Jana dongakan penembakan pseudorawak di antara -40^o dan $+40^o$.
    c. Tentukan *profile* trajektori peluru dimana lebar dan ketinggiannya diketahui sebagai 0.2^o dan 1^o masing-masing.
    d. Simpan *profile* trajektori peluru.
    e. Guna struktur data *bucket* untuk menentukan mana satu rakan, jika ada, mungkin terkena tembakan itu.
    f. Bagi setiap rakan yang mungkin terkena tembakan itu:
        i. Simpan identiti rakan tersebut.
5. Rekod masa tamat (t_e)
6. Bagi setiap trajektori yang dijana:
    a. Paparkan *profile* trajektori tersebut.
    b. Bagi setiap rakan yang mungkin terkena trajektori itu:
        i. Paparkan *profile* rakan tersebut.
7. Hitung dan paparkan purata masa per trajektori, $(t_e - t_s)/1000$.

Walaupun *profile* rakan dijana dalam units θx, θy, θh dan θw, ini mungkin bukan nilai-nilai yang paling mudah digunakan bagi tujuan menentukan persilangan di antara *profile* trajektori dan *profile* rakan. Prototaip itu tidak perlu menyimpan θx, θy, θh dan θw. Ia boleh menyimpan *profile* dalam bentuk yang lebih sesuai bagi penentuan persilangan *profile*.

3 PENGELAP CERMIN PENAHAN ANGIN

Dalam usaha mengurangkan bilangan alat mekanik dalam satu mekanisme pengelap cermin penahan angin, satu penghasil automobil berhasrat untuk menggantikan mekanisme motor DC semasanya dengan mekanisme motor pelangkah tanpa-berus.

Secara konsep, motor pelangkah terdiri daripada sebuah magnet tetap bercagak pada satu gelendong dengan 2 gelung dawai medan magnet yang boleh diberi tenaga secara bersiri untuk memandu gelendong pada satu atau lain-lain haluan. Gambar rajah di bawah menunjukkan mekanisme motor pelangkah tersebut:



Apabila suis A disambungkan ke +12V dan suis B disambungkan ke bumi (iaitu 0V), arus biasa akan mengalir dari A ke B. Ini akan menyebabkan teras dengan gelung dawai A-B dimagnetkan supaya bahagian atasnya menjadi kutub utara dan bahagian bawahnya kutub selatan. Ini seterusnya akan mengakibatkan gelendong itu berputar mengikut arah jam sehingga kutub utara gelendong berada di bahagian bawah. Dengan mengoperasikan suis A ke suis D dalam tertib yang betul, gelendong boleh berputar dalam salah satu arah.

Suis A ke D adalah dikawal menerusi Register Kawalan Suis 4-bit yang dikenali sebagai sw*itch control register (SCR)* dalam bahasa Inggeris. Ia boleh ditetapkan oleh perisian. Bit bererti terkecil bagi SCR mengoperasikan suis A dan bit paling bererti mengendalikan suis D. Apabila bit SCR ialah '0', suis yang berkenaan akan disambung ke bumi; apabila bit ialah '1', suis akan disambung ke +12V.

SCR boleh ditetapkan menerusi C atau C++ dengan memanggil fungsi berikut:

```
void SetSCR (int scrVal)
```

SCR boleh ditetapkan menerusi Java dengan memanggil kaedah statik yang berikut:

```
class WiperControl {
     //...
     public void setSCR (int scrVal)
     //...
}
```

Bagi kesemua bahasa pengaturcaraan C, C++ dan Java, `scrVal` ialah nilai yang dimuatkan ke dalam SCR. Hanya 4 bit bererti terkecil bagi `scrVal` dimuatkan ke dalam SCR. Bit-bit tertib lebih tinggi adalah diabaikan.

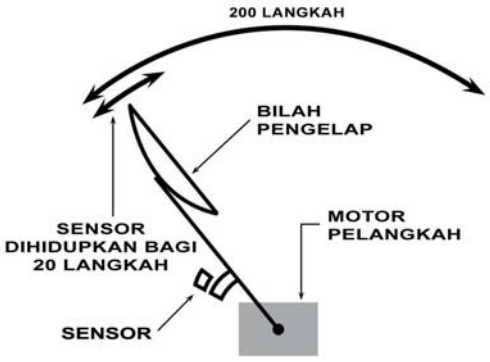
Untuk memusing gelendong itu, gelung-gelung dawai mesti diberi tenaga mengikut jadual yang berikut sama ada dalam jujukan menaik atau menurun. Jika gelung-gelung

dawai itu diberi tenaga dalam jujukan menurun, motor itu akan berputar mengikut arah jam. Jika mereka diberi tenaga dalam jujukan menaik, motor itu pula akan berputar mengikut arah lawan jam.

Untuk putaran ikut arah jam	**NILAI SCR**				**Untuk putaran arah lawan jam**
	D	**C**	**B**	**A**	
↓	0	0	0	1	↑
	0	1	0	0	
	0	0	1	0	
	1	0	0	0	

Setiap baris di dalam jadual di atas dirujuk sebagai satu ‘langkah’. Dalam mekanisme pengelap cermin penahan angin itu, motor pelangkah adalah digearkan supaya pergerakan bilah pengelap dari posisi paling kiri ke posisi paling kanan melibatkan 200 langkah. Mekanisme pengearan itu adalah tidak menentang arah motor pelangkah. Jika motor pelangkah berputar mengikut arah jam, maka bilah akan bergerak dari kiri ke kanan. Sebaliknya, bilah pengelap akan bergerak dari kanan ke kiri.

Terdapat satu sensor yang dapat mengesan posisi bilah pengelap cermin apabila bilah itu berhampiran dengan had paling kiri pergerakannya. Gambar rajah berikut menunjukkan bagaimana sensor tersebut bertugas.



Apabila bilah pengelap mengayun dari kanan ke kiri, sensor itu akan dihidupkan. Bilah pengelap mesti berhenti and menterbalikkan arah ayunannya 20 langkah selepas peralihan ini.

Keadaan sensor boleh dikesan daripada C atau C++ dengan memanggil fungsi yang berikut:

```
int GetSensorState()
```

`nilai pulangan`
ialah sifar jika sensor dipadamkan, atau bukan sifar jika sensor dihidupkan.

Ia boleh dikesan dalam Java dengan memanggil:

```
Class WiperControl {
     //...
     public boolean getSensorState()
     //...
}
```

`nilai pulangan`
ialah palsu jika sensor dipadamkan, atau benar jika sensor dihidupkan.

Kelajuan operasi biasa untuk motor itu ialah 200 langkah sesaat, tetapi jika ia terus dihidupkan daripada keadaan berhenti kepada 200 langkah sesaat, ia hampir pasti akan melangkau beberapa langkah dan pecutan yang dihasilkan sedemikian akan merosakkan gear pengurangan dan mekanisme bilah pengelap.

Had maksimum pecutan yang boleh diterima ialah 1,250 langkah per saat persegi mengikut mana mana satu arah. Peningkatan kelajuan ini serta pengurangannya dicapai, dalam lingkungan had yang dikehendaki oleh mekanisme, dengan menggunakan siri masa antara-langkah yang tersenarai di jadual di bawah apabila pergerakan bilah pengelap dihidupkan, dimatikan dan disongsangkan haluannya.

Masa Antara-Langkah (milisaat)															
42	17	13	11	10	9	8	8	7	7	6	6	6	6	6	5

Tugas anda adalah menulis satu program komputer untuk mengawal motor pelangkah bagi pengelap cermin penahan angin itu. Program ini boleh ditulis dalam bahasa pengaturcaraan C, C++ atau Java.

Jika anda menulis dalam bahasa pengaturcaraan C atau C++, program anda mestilah mengandungi fungsi-fungsi berikut yang akan dipanggil oleh sistem pengendalian mikropemproses.

```
void Initialise ();
void StartWiping ();
void StopWiping ();
void Tick();
```

Jika anda menulis dalam bahasa pengaturcaraan Java, program anda mesti melaksanakan antara muka yang berikut:

```
interface WiperInterface {
     void initialise(WiperControl control);
     void startWiping ();
     void stopWiping ();
     void tick ();
}
```

Dalam kesemua bahasa pengaturcaraan, fungsi `Initialise` mesti mematikan suis bagi semua gelung dawai dahulu dengan menetapkan SCR kepada sifar. Kemudian, ia mesti memberi nilai permulaan kepada semua struktur data dalaman dan menguji keadaan sensor. Jika sensor itu sudah dipadamkan, ia mesti memulakan satu proses untuk mengembalikan bilah pengelap ke posisi permulaan yang paling kiri pada kelajuan tanpa-progresif maksimum 42ms selangkah. Apabila bilah berada di posisi permulaan paling kiri, program pengawal mesti mematikan suis bagi gelung-gelung dawai medan magnet dengan menetapkan SCR kepada sifar. Jika sensor itu telah dihidupkan apabila `Initialise` dipanggil, bilah pengelap mestilah tetap berada di posisi yang awal.

Versi Java bagi kaedah `initialise` menerima satu rujukan kepada *instance* kelas `WiperControl` yang boleh digunakan untuk membaca keadaan sensor dan menetapkan SCR.

Fungsi `StartWiping` mesti memulakan pergerakan ulang-alik bilah pengelap. Fungsi `StartWiping` boleh dipanggil bila-bila masa selepas `Initialise` dipanggil. Jika fungsi `StartWiping` dipanggil semasa bilah itu sedang dikembalikan ke posisi permulaan paling kiri dengan 42ms selangkah, proses kembali itu mesti dihabiskan dahulu sbelum kitaran ulang-alik biasa dimulakan. Sebaik saja bilah pengelap berada pada posisi permulaan paling kiri, program tu mesti meningkatkan kelajuannya ke nilai maksimum dalam arah kiri-ke-kanan mengikut jadual kelajuaan. Kemudian, program itu mesti mengurangkan kelajuan bilah itu supaya ia berhenti 180 langkah selepas sensor dipadamkan. Selepas itu, ia mesti menyongsangkan arah pusingan motor, menambah kelajuan bilah, melaraskan lokasi bilah berdasarkan teori apabila sensor dihidupkan semula, mengurangkan kelajuan dan menyongsangkan arah putaran motor itu dan sebagainya sehingga fungsi `StopWiping` dipanggil.

Fungsi `StopWiping` boleh dipanggil bila-bila masa selepas `Initialise` dipanggil, walaupun bilah itu sudah pegun. Dalam semua kes, bilah pengelap mesti dibenarkan untuk kembali ke posisi permulaan paling kiri secara normal dengan kitaran yang sedang berlangsung apabila `StopWiping` dipanggil. Gelung-gelung dawai motor itu mestilah kemudian dimatikan suis mereka dengan melaraskan SCR kepada sifar.

Fungsi `Tick` adalah dipanggi oleh system pengendalian sekali setiap milisaat.

Program pengawal itu tidak boleh mengunci kawalan dalam mana-mana fungsi aplikasi. Jika kawalan kekal dalam fungsi `StartWiping,` maka fungsi `StopWiping` tidak akan dapat dipanggil.

4 FAKTORIAL-25

Tuliskan satu program untuk menghitung dan memaparkan nilai tepat bagi faktorial-25 (iaitu 25 * 24 * 23 * ... * 3 * 2) tanpa bantuan daripada sebarang fungsi aritmetik kepersisan teperluas yang terdapat dalam bahasa pengatucaraan atau sistem pengendalian.

Faktorial-25 adalah dalam tertib 2^{84} atau 10^{26}. C dan C++ mempunyai *32-bit long integers* dan Java mempunyai *64-bit long integers*, kesemua ini tidak mempunyai ketepatan yang cukup untuk melaksanakan pengiraan yang dikehendaki. Dalam semua bahasa pengaturcaraan, ketepatan maksimum jenis data titik-apung adalah dalam tertib 52 bits.

Anda harus mempertimbangkan dengan teliti bagaimana caranya nombor kepersisan teperluas itu patut distorkan. Sesetengah format storan akan menyebabkan tugas itu menjadi agak mudah. Secara perbandingan, format yang lain akan melibatkan usaha lebih daripada dua kali ganda.