E-GENTING
PROGRAMMING COMPETITION 2017

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.	Pathfinding	300
2.	Search Results	200
3.	Image Scaling	400
4.	Unverified Report	200

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 PATHFINDING

An online game operator called Guessing Games (GG) has a new game that has recently become very popular among your friends.

The game has a red, a blue and a green ball that will each travel down a different tube to the bottom, and exit the tubes. The middle sections of the tubes, where they crisscross exactly 5 times, are covered at first to hide where the tubes go. The goal of the game is to guess where each ball will exit the tubes. After all guesses are received, the game reveals the crisscross level by level. Figure 1 shows the start and end of the game.

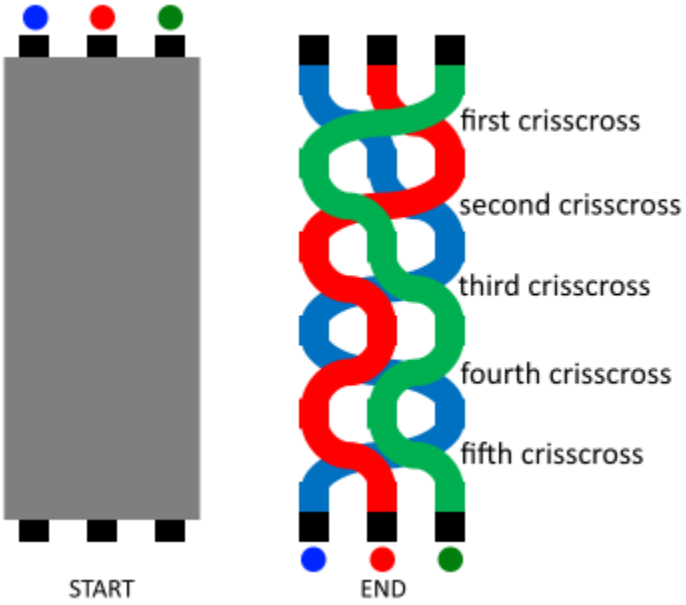


Figure 1: Sample of Start and End of the Game

This coming Saturday, five of your friends will be throwing a party to play the game together but they will not let you play. They have decided to make you the judge and force you to count their scores for the 50 rounds that they will be playing.

You have decided to prank your friends by creating a rigged clone of the game. In your rigged version, the game will always make the balls exit in such a way that none of your friends can win.

In order to make the rigged game, you need to find a way to determine the path of the balls to the desired result. You have decided to write that function first.

The function must receive the starting location of the balls, and the ending locations your friends selected for the balls.

The function must return the paths taken by the balls. The result is an array of three paths. A path is represented by an array of five locations. Each location is the position of the ball after a crisscross. Each time the tubes crisscross, at least two tubes will swap positions.

To prevent your friends from identifying that the game is rigged, the function must randomly select the paths, so that there is very little chance of two rounds producing the same paths.

You may use the declarations shown in Figure 2, or write your own declarations in the language of your choice. Your function must at least pass a test program similar to the one shown in

Figure 3.

```
// C++
enum Location_t {
    LOC_LEFT,
    LOC_MIDDLE,
    LOC_RIGHT
};
enum Ball_t {
    BALL_RED,
    BALL_GREEN,
    BALL_BLUE
};

void
FindPath (
    Location_t      path[3][5],             // Array of path taken by balls
    Location_t      start[3],               // Starting location of balls
    Location_t      guess[5][3])            // Guesses made by friends
{
    // TODO:
}
```

```
// Java
class Game_c {
    enum Location_t {
        LOC_LEFT,
        LOC_MIDDLE,
        LOC_RIGHT,
    };

    public static final int BALL_RED = 0;
    public static final int BALL_GREEN = 1;
    public static final int BALL_BLUE = 2;

    static void
    FindPath (
        Location_t      path[][],           // Array of path taken by balls
        Location_t      start[],            // Starting location of balls
        Location_t      guess[][])          // Guesses made by friends
    {
        // TODO:
    }
}
```

```
// C#
class Game_c {
    enum Location_t {
        LOC_LEFT,
        LOC_MIDDLE,
        LOC_RIGHT,
    };

    public const int BALL_RED = 0;
    public const int BALL_GREEN = 1;
    public const int BALL_BLUE = 2;

    static void
    FindPath (
        Location_t[,]   path,               // Array of path taken by balls
        Location_t[]    start,              // Starting location of balls
        Location_t[,]   guess)              // Guesses made by friends
    {
        // TODO:
    }
}
```

Figure 2: Sample Declaration for Function to Find Path

```
// C++
#include <iostream>
#include <cstring>
using namespace std;
#include <PathFind.h>

int
main ()
{
    Location_t          path[3][5];         // Array of path taken by balls
    int                 lvl;                // Crisscross level
    int                 pl;                 // Player number
    char                buf[512];           // Formatting buffer

    const Location_t    start[] =           // Starting location of balls
        {LOC_LEFT, LOC_MIDDLE, LOC_RIGHT};
    const Location_t        guess[][] = {   // Guesses made by friends
        {LOC_LEFT, LOC_MIDDLE, LOC_RIGHT},
        {LOC_LEFT, LOC_RIGHT, LOC_MIDDLE},
        {LOC_MIDDLE, LOC_LEFT, LOC_RIGHT},
        {LOC_MIDDLE, LOC_RIGHT, LOC_LEFT},
        {LOC_RIGHT, LOC_LEFT, LOC_MIDDLE}
    };

    // Test the function

    FindPath (path, start, guess);

    // Show the result

    strcpy (buf, "R G B");
    cout << buf << '\n';
    for (lvl = 0; lvl < 5; lvl++) {
        buf[path[BALL_RED][lvl] * 2] = 'R';
        buf[path[BALL_GREEN][lvl] * 2] = 'G';
        buf[path[BALL_BLUE][lvl] * 2] = 'B';
        cout << buf << '\n';
    }

    // Verify that no winner is found

    for (pl = 0; pl < 5; pl++) {
        if (
            guess[pl][BALL_RED] == path[BALL_RED][LVL_CNT-1] &&
            guess[pl][BALL_GREEN] == path[BALL_GREEN][LVL_CNT-1] &&
            guess[pl][BALL_BLUE] == path[BALL_BLUE][LVL_CNT-1]
        )
            cout << "** Player " << pl << " wins!\n";
    }
    return 0;
}
```

```
// Java
class Game_c {
    public static void
    main (
        Location_t      args[],             // Starting location of balls
    {
        Location_t      path[][];           // Array of path taken by balls
        int             lvl;                // Crisscross level
        int             pl;                 // Player number
        StringBuilder   str;                // String builder

        final Location_t start[] =          // Starting location of balls
            {Location_t.LOC_LEFT, Location_t.LOC_MIDDLE,
                                                Location_t.LOC_RIGHT};
        final Location_t guess[][] = {      // Guesses made by friends
            {Location_t.LOC_LEFT, Location_t.LOC_MIDDLE,
                                                Location_t.LOC_RIGHT},
            {Location_t.LOC_LEFT, Location_t.LOC_RIGHT,
                                                Location_t.LOC_MIDDLE},
            {Location_t.LOC_MIDDLE, Location_t.LOC_LEFT,
                                                Location_t.LOC_RIGHT},
            {Location_t.LOC_MIDDLE, Location_t.LOC_RIGHT,
                                                Location_t.LOC_LEFT},
            {Location_t.LOC_RIGHT, Location_t.LOC_LEFT,
                                                Location_t.LOC_MIDDLE}
        };

        // Test the function

        path = new Location_t[3][5];
        FindPath (path, start, guess);

        // Show the result

        str = new StringBuilder ("R G B");
        System.out.println (str.toString());
        for (lvl = 0; lvl < 5; lvl++) {
            str.setCharAt(path[BALL_RED][lvl].ordinal() * 2, 'R');
            str.setCharAt(path[BALL_GREEN][lvl].ordinal() * 2, 'G');
            str.setCharAt(path[BALL_BLUE][lvl].ordinal() * 2, 'B');
            System.out.println (str.toString());
        }

        // Verify that no winner is found

        for (pl = 0; pl < 5; pl++) {
            if (
                guess[pl][BALL_RED] == path[BALL_RED][LVL_CNT-1] &&
                guess[pl][BALL_GREEN] == path[BALL_GREEN][LVL_CNT-1] &&
                guess[pl][BALL_BLUE] == path[BALL_BLUE][LVL_CNT-1]
            ) {
                System.out.print ("** Player ");
                System.out.print (pl.toString());
                System.out.println (" wins!");
            }
        }
    }
}
```

```
// C#
using System;
using System.Text;

class Game_c {
    static void
    Main (
        string[]        args)               // Program arguments
    {
        Location_t[][]  path;               // Array of path taken by balls
        int             lvl;                // Crisscross level
        int             pl;                 // Player number
        StringBuilder   str;                // String builder

        Location_t[] start =                // Starting location of balls
            {Location_t.LOC_LEFT, Location_t.LOC_MIDDLE,
                                                Location_t.LOC_RIGHT};
        Location_t[,] guess = {             // Guesses made by friends
            {Location_t.LOC_LEFT, Location_t.LOC_MIDDLE,
                                                Location_t.LOC_RIGHT},
            {Location_t.LOC_LEFT, Location_t.LOC_RIGHT,
                                                Location_t.LOC_MIDDLE},
            {Location_t.LOC_MIDDLE, Location_t.LOC_LEFT,
                                                Location_t.LOC_RIGHT},
            {Location_t.LOC_MIDDLE, Location_t.LOC_RIGHT,
                                                Location_t.LOC_LEFT},
            {Location_t.LOC_RIGHT, Location_t.LOC_LEFT,
                                                Location_t.LOC_MIDDLE}
        };

        // Test the function

        path = new Location_t[3, 5];
        FindPath (path, start, guess);

        // Show the result

        str = new StringBuilder ("R G B");
        Console.WriteLine(str.ToString());
        for (lvl = 0; lvl < 5; lvl++) {
            str[(int)path[BALL_RED, lvl] * 2] = 'R';
            str[(int)path[BALL_GREEN, lvl] * 2] = 'G';
            str[(int)path[BALL_BLUE, lvl] * 2] = 'B';
            Console.WriteLine(str.ToString());
        }

        // Verify that no winner is found

        for (pl = 0; pl < 5; pl++) {
            if (
                guess[pl][BALL_RED] == path[BALL_RED][LVL_CNT-1] &&
                guess[pl][BALL_GREEN] == path[BALL_GREEN][LVL_CNT-1] &&
                guess[pl][BALL_BLUE] == path[BALL_BLUE][LVL_CNT-1]
            )
                Console.WriteLine("** Player " + pl.ToString() + " wins!");
        }
    }
}
```

Figure 3: Sample Test Program

2 SEARCH RESULTS

You have just joined a company that recently added a command bar to its most popular desktop program. A screenshot of the command bar is shown in Figure 4. The command bar accepts a 6-character case-sensitive text command as shortcut to a particular screen.

Figure 4: Screenshot of Command Bar

The command bar also displays the description of the command that the user has typed. However, displaying the description of the current command is not particularly useful, especially for a user who has forgotten the exact command that he needs.

Being the proactive programmer that you are, you propose that the command bar should instead search for commands containing the text the user has typed and display them on the bar.

Management agrees with your idea but also raises additional requirements.

The command bar must search both the command and its description. When searching, the command bar must ignore case differences in the description. Both the command and its description only contain ASCII characters.

If the search results cannot fit on the command bar, the command bar must separate the search results into multiple pages.

For each command, the command bar must display the command and its description in a single line on the same page. The command bar must display the colon ‘:’ character between the command and its description. The command bar must display a pipe ‘|’ character between a command description and the next command. If a command and its description cannot fit into the available space, it should be placed on the next page. However, the command bar must display at least one command and description pair on each page.

The command bar displays text in fixed-width font but the window containing the command bar is resizeable. Therefore, the number of commands per page and the number of pages may vary. When the command bar is resized, it should move commands to the pages based on the updated capacity. If the requested page number is more than the number of pages used by the search result, the command bar should display the last page.

The command bar must display the commands in priority order. It is also possible that the list of commands may change based on the access rights of the currently logged in user.

You set forth to fulfill those requirements by first writing a function to the search the command list and produce the search result. Your function must accept the following parameters:

1. search word;
2. current page number;
3. a list of command name and description pairs that is sorted by priority;
4. buffer capacity as determined by the available space on the command bar;
5. buffer to store the search result.

Figure 5 and Figure 6 shows example declaration of the function in C++.

```
// CommandItem.h

struct CmdItem_t {
    char            cmdItemCmd[6];      // Command
    char            cmdItemDesc[256];   // Description
};
```

Figure 5: Command Item Structure Declaration in C++

```
// CmdBar.h - Command Bar

class CmdBar_c {
public:

    // Perform search and return result for a particular page

    void CmdBarSearch (
        const char      *word,          // Search word
        int             page,           // Current page number
        const CmdItem_t *itemArr,       // Pointer to items
        size_t          itemCnt,        // Number of items
        size_t          cap,            // Buffer capacity
        char            *result)        // Buffer to store result
    {
        // TODO:
    }
};
```

Figure 6: Command Bar Class Implementation in C++

3 IMAGE SCALING

The Peephole operating system stores images in the 24-bit colour format described below. Peephole has functions to capture and display images, but has no provision for even the simplest cropping, expansion or contraction of stored images. Your task is to write a function that receives a stored image in the 24-bit Peephole format, and parameters for cropping and expanding or contracting the image, and emits an image that is cropped and expanded or contracted in accordance with the parameters.

The 24-bit Peephole format consists of a header followed by an array of pixel colour structures. The header has the structure represented in Table 1.

Table 1 – 24-bit Peephole Image Format

Offset	Contents			
0 to 3	'P'	'H'	'2'	'4'
4 to 7	width			
8 to 11	height			
12 to 14	red[0,0]	green[0,0]	blue[0,0]	
15 to 17	red[1,0]	green[1,0]	blue[1,0]	
: :	:	:	:	
: :	red[n-1,m-1]	green[n-1,m-1]	blue[n-1,m-1]	

The first 4 four bytes of the image structure identify the structure and contain the characters 'P', 'H', '2' and '4' in that order.

The next four bytes contain the width of the image in pixels and this is followed by another four bytes that contain the height of the image in pixels. Both the width and height are in big endian format. I.e. the first byte contains the most significant 8 bits of the value and the last byte contains the least significant 8 bits of the value. For example, if the image width was 6500 pixels, then the four width bytes could contain 0, 0, 25 and 100 in that order.

The width and height are followed by an array of 3-byte structures that contain the red, green and blue intensities of each pixel. The pixels are ordered by row and then column in a right-handed co-ordinate system. For example, if the image had two rows and three columns, the rows and columns would be numbered as shown in Figure and stored in the order presented in Table 1.

[0,1]	[1,1]	[2,1]
[0,0]	[1,0]	[2,0]

Figure 7 - Pixel Numbering Convention

Table 2 – Pixel Order Example

red[0,0]	green[0,0]	blue[0,0]
red[1,0]	green[1,0]	blue[1,0]
red[2,0]	green[2,0]	blue[2,0]
red[0,1]	green[0,1]	blue[0,1]
red[1,1]	green[1,1]	blue[1,1]
red[2,1]	green[2,1]	blue[2,1]

If you program in C or C++, your function must have the declaration below:

```
unsigned char *ScaleImage (
     const unsigned char *unscaledImage,
     long                xOfs,
     long                yOfs,
     unsigned long       unscaledWidth,
     unsigned long       unscaledHeight,
     unsigned long       scaledWidth,
     unsigned long       scaledHeight,
     unsigned char       fillRed,
     unsigned char       fillGreen,
     unsigned char       fillBlue);
```

`unscaledImage`
: is a pointer to an array of bytes that contain the unscaled image in the Peephole 24-bit format.

`xOfs`
: is the horizontal offset from the bottom left corner of the unscaled image to the left edge of the crop rectangle in pixels. `xOfs` may be negative.

`yOfs`
: is the vertical offset from the bottom left corner of the unscaled image to the bottom edge of the crop rectangle pixels. `yOfs` may be negative.

`unscaledWidth`
: is the width of the crop rectangle in pixels.

`unscaledHeight`
: is the height of the crop rectangle in pixels.

`scaledWidth`
: is the width of the scaled image in pixels.

`scaledHeight`
: is the height of the scaled image in pixels.

`fillRed`
: is the red intensity of the fill colour (0 to 255).

`fillGreen`
: is the green intensity of the fill colour (0 to 255).

`fillBlue`
: is the blue intensity of the fill colour (0 to 255).

`Return value`
: is a pointer to an array of bytes in dynamic memory (i.e. memory allocated with either the **new** operator or **malloc** function) that contains the scaled image.

`ScaleImage` must stretch or compress the crop rectangle represented by the `xOfs`, `yOfs`, `unscaledWidth` and `unscaledHeight` parameters into a new image having dimensions `scaledWidth` and `scaledHeight` and return a pointer to the new image, which must be stored in dynamically allocated memory.

The crop rectangle may not lie wholly within the bounds of the unscaled image, in which case, `ScaleImage` must assign the fill colour, as defined by the `fillRed`, `fillGreen` and `fillBlue` parameters, to pixels in the crop rectangle that are outside the bounds of the unscaled image.

`ScaleImage` must assign each pixel in the scaled image the overlap-area-weighted average colour of the pixels in the crop rectangle that the pixel in the scaled image overlaps.

For example, consider the 4 x 3 crop rectangle (in red dashes) and 3 x 2 scaled image (in blue) in Figure 8

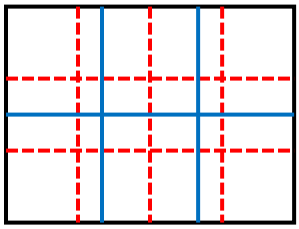


Figure 8 – 4 x 3 Crop Rectangle and 3 x 2 Scaled Image

In this case the pixel [1,0] in the scaled image overlaps pixels [1,0], [2,0], [1,1] and [2,1] of the crop rectangle and the overlap areas are 2/6, 2/6, 1/6 and 1/6 respectively, so the red intensity of pixel [1,0] in the scaled image would be:

```
(2*red[1,0] + 2*red[2,0] + red[1,1] + red[2,1]) / 6
```

When calculating pixel intensities, `ScaleImage` must round the rational number given by the overlap-area-weighted average colour to the nearest integer. I.e. if the red

intensity given by above formula was 155/6, `ScaleImage` must round the intensity to 26 and if the intensity given by the above formula was 151/6, it must round the intensity to 25. If the theoretical intensity is exactly midway between two integers, for example, 153/6, `ScaleImage` may round the theoretical intensity either up or down to the nearest integer.

Floating point arithmetic under the Peephole operating system is neither fast nor precise. `ScaleImage` should not use floating point arithmetic.

As mentioned previously, the Peephole operating system does not support image cropping or scaling, so any language-based functions that would ordinarily be available for image manipulation are not available in the Peephole implementation of the language and `ScaleImage` must not attempt to use them.

4 UNVERIFIED REPORT

A company called Order Me Around (OMA) operates an online food ordering system. The system receives food orders from mobile apps and web browsers of users scattered throughout the country.

When the system receives an order, it selects the few best restaurants based on proximity to the user, order load at the restaurants and user ratings for the restaurants. The system relays the order to the selected restaurants. The restaurants may choose to accept or decline the order. The first restaurant to accept the order gets the job to handle the order and deliver the food. The system allows restaurants to change their mind. If the restaurant with the job declines the order, the system immediately reassigns the order to the next restaurant that accepts the order.

Sometimes, customers complain that they never receive their orders and request for refunds. There is no way to prove whether the food was delivered or not. In most cases, OMA has no choice but to refund the user as well as pay the restaurant for the service, to maintain good relationships with the users and the restaurants.

In order to fix this problem and prevent further losses, OMA decided to implement a verification procedure. When an order is placed, the system provides the user with a secret code. The user must provide the secret code to the delivery person to receive his food. OMA will provide the delivery person with a new mobile app that can verify the validity of the secret code. If the secret code is not verified, OMA will not release the payment from the customer to the restaurant.

As part of the development team, your job is to write a program that generates a report containing the list of transactions in which the secret code was not verified. The report must show the following information in a layout similar to the one shown in Figure 7:

1. for each restaurant (sorted alphabetically):
    a. restaurant name;
    b. for each order fufilled but not verified (sorted by order number):
        i. order number;
        ii. order value;
    c. total order value for orders fulfilled but not verified;
    d. number of orders verified;
    e. percentage of orders verified over orders fulfilled (rounded or truncated to 2 decimal places);
2. grand total order value for orders fulfilled but not verified;
3. percentage of total orders verified over total orders fulfilled (rounded or truncated to 2 decimal places).

The system contains the database tables shown in Figure 8. Time is stored in units of seconds since midnight on 1st January 1970. Value is stored in integer cent values. For example value 1.0 is 1 cent, and 100.0 is RM 1.

```
======== UNVERIFIED ORDERS REPORT ========

RESTAURANT           ORDER NO      VALUE VERIFIED   (%)
-------------------- -------- ---------- ------ -------
KL-FG-CD               334567     120.40
                       334656     124.95
                              ----------
TOTAL                             245.35      8   80.00
                              ----------

MR DUNELD'S            334455     320.00
                       334456      24.95
                              ----------
TOTAL                             344.95     18   95.00
                              ----------

POPEARS                                      10  100.00

-------------------- -------- ---------- ------ -------
GRAND TOTAL                       590.30     36   90.00
-------------------- -------- ---------- ------ -------
```

Figure 7: Sample Report

```
// Customer order

exec sql create table ord (
    ordOrderNo      integer not null,           // Order number
    ordTim          double precision not null,  // Order time
    ordValue        integer not null,           // Order value
    ordVerified     smallint not null,          // Verified flag (0=N,1=Y)
);
exec sql create unique cluster index ordOrderNoInd on
    ord (ordOrderNo);
```

```
// Order response from restaurants

exec sql create table ordRsp (
    ordRspOrderNo   integer not null,           // Order number
    ordRspRestId    integer not null,           // Restaurant id
    ordRspTim       double precision,           // Response time
    ordRspAccept    smallint                    // Accept flag (0=N,1=Y)
);
exec sql create index ordRspOrderNoInd on
    ordRsp (ordRspOrderNo, ordRspRestId, ordRspTim);
```

```
// Restaurant record

exec sql create table res (
    resRestId       integer not null,           // Restaurant id
    resRestName     char(21) not null           // Restaurant name
);
exec sql create unique cluster index resRestIdInd on
    res (resRestId);
```

Figure 8: Database Schema

Table 3 shows sample records in the order response table. In the sample data, restaurant 4 fulfilled order 121277 because restaurant 2 declined the order after being the first to accept the order. Retaurant 6 fulfilled order 121278, while restaurant 4 fulfilled order 121279.

Table 3: Sample Data in Order Response Table

Order Number	Restaurant ID	Response Time	Accept Flag
121277	2	1510484400	1
121278	6	1510484410	1
121277	2	1510484445	0
121279	4	1510484447	1
121277	4	1510484450	1

PERTANDINGAN PENGATURCARAAN E-GENTING 2017

Arahan-arahan am:

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

No	Tajuk	Markah
1.	Pencarian Laluan	300
2.	Keputusan Carian	200
3.	Penskalaan Gambar	400
4.	Laporan Pesanan Yang Tidak Dikesahkan	200

10. Kecuali dinyatakan, program anda boleh ditulis dengan menggunakan mana-mana bahasa pengaturcaraan utama di bawah mana-mana sistem operasi utama.
11. Kecuali dinyatakan, anda boleh menggunakan semua fungsi piawaian perpustakaan dalam bahasa pengaturcaraan dan sistem operasi yang anda pilih.
12. Perkataan-perkataan ‘must’, ‘must not’, ‘required’, ‘should’, ‘should not’, dan ‘may’ adalah ditafsirkan seperti diterangkan dalam RFC 2119[1].
13. Anda TIDAK perlu menjawab semua soalan.

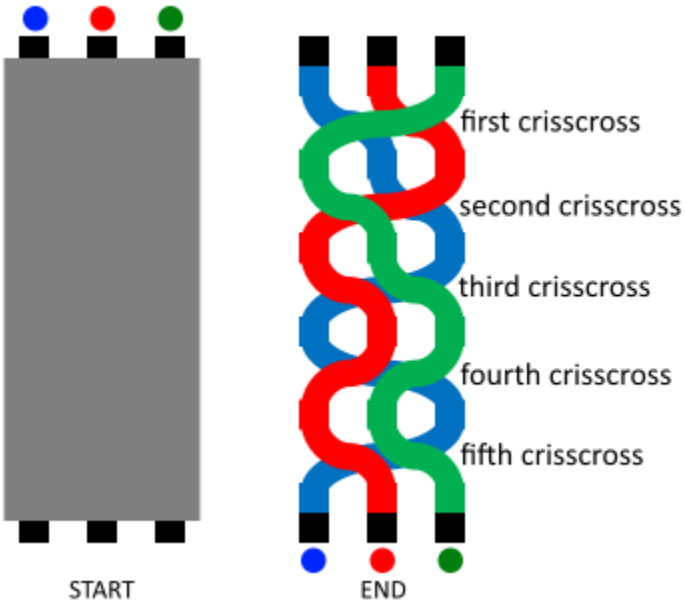
---

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

1 PENCARIAN LALUAN

Satu operator permainan dalam talian yang bernama Guessing Games (GG) mempunyai sebuah permainan baru yang sangat popular baru-baru ini di kalangan kawan anda.

Permainan tersebut mempunyai sebiji bola merah, sebiji bola biru dan sebiji bola hijau. Tiap-tiap bola tersebut akan melalui tiub yang berbeza ke bahagian bawah dan keluar dari tiub-tiub tersebut. Bahagian tengah tiub-tiub tersebut, yang saling melintasi sebanyak 5 kali, tertutup pada awal permainan untuk menyembunyikan arah tujuan tiub-tiub tersebut. Matlamat permainan tersebut ialah untuk meneka di mana setiap bola akan keluar dari tiub-tiub. Selepas semua tekaan diterima, permainan tersebut akan mendedahkan lintasan tersebut tahap demi tahap. Gambarajah 1 menunjukkan permulaan dan penamatan permainan tersebut.



Gambarajah 1: Contoh Permulaan dan Penamatan Permainan Tersebut

Pada hari Sabtu yang akan datang, 5 orang kawan anda akan mengadakan satu parti untuk bermain permainan tersebut tetapi mereka tidak membenarkan anda bermain bersama. Mereka mengambil keputusan untuk melantik anda sebagai hakim dan memaksa anda mengira markah mereka untuk 50 pusingan yang mereka akan bermain.

Anda mengambil keputusan untuk bergurau dengan kawan-kawan anda dengan membina permainan klon tersebut yang bersifat tidak adil. Dalam permainan versi anda, permainan tersebut akan memastikan tiada seorang daripada kawan-kawan anda boleh menang, kerana lokasi penamatan bola-bola tidak akan menepati tekaan kawan-kawan anda.

Untuk menghasilkan permainan yang tidak adil ini, anda perlu menentukan laluan bola-bola untuk menghasilkan keputusan yang dikehendaki. Anda mengambil keputusan untuk mengaturcara fungsi tersebut dahulu.

Fungsi tersebut mesti menerima lokasi permulaan bola-bola dan lokasi penamatan yang kawan-kawan anda telah pilih untuk bola-bola tersebut.

Fungsi tersebut mesti memulangkan laluan yang dilalui oleh bola-bola tersebut. Keputusan tersebut merupakan satu tatasusunan yang mengandungi tiga laluan. Satu laluan diwakili oleh satu tatasusunan yang mempunyai 5 lokasi. Setiap lokasi ialah kedudukan bola tersebut selepas satu lintasan. Setiap kali tiub-tiub saling melintasi, sekurang-kurangnya 2 tiub akan saling bertukar kedudukan.

Untuk mengelakkan kawan-kawan anda mengetahui bahawa permainan tersebut telah diubahsuai, fungsi tersebut mesti memilih laluan-laluan secara rawak, untuk mengurangkan kemungkinan di mana dua pusingan menghasilkan laluan yang sama.

Anda boleh menggunakan pengisytiharan seperti dalam Gambarajah 2, atau menulis pengisytiharan anda dalam bahasa pengaturcaraan pilihan anda. Fungsi anda mesti sekurang-kurangnya lulus ujian program seperti dalam Gambarajah 3.

```
// C++
enum Location_t {
    LOC_LEFT,
    LOC_MIDDLE,
    LOC_RIGHT
};
enum Ball_t {
    BALL_RED,
    BALL_GREEN,
    BALL_BLUE
};

void
FindPath (
    Location_t      path[3][5],             // Array of path taken by balls
    Location_t      start[3],               // Starting location of balls
    Location_t      guess[5][3])            // Guesses made by friends
{
    // TODO:
}
```

```
// Java
class Game_c {
    enum Location_t {
        LOC_LEFT,
        LOC_MIDDLE,
        LOC_RIGHT,
    };

    public static final int BALL_RED = 0;
    public static final int BALL_GREEN = 1;
    public static final int BALL_BLUE = 2;

    static void
    FindPath (
        Location_t      path[][],           // Array of path taken by balls
        Location_t      start[],            // Starting location of balls
        Location_t      guess[][])          // Guesses made by friends
    {
        // TODO:
    }
}
```

```
// C#
class Game_c {
    enum Location_t {
        LOC_LEFT,
        LOC_MIDDLE,
        LOC_RIGHT,
    };

    public const int BALL_RED = 0;
    public const int BALL_GREEN = 1;
    public const int BALL_BLUE = 2;

    static void
    FindPath (
        Location_t[,]   path,               // Array of path taken by balls
        Location_t[]    start,              // Starting location of balls
        Location_t[,]   guess)              // Guesses made by friends
    {
        // TODO:
    }
}
```

Gambarajah 2: Contoh Pengisytiharan untuk Fungsi Carian Laluan

```
// C++
#include <iostream>
#include <cstring>
using namespace std;
#include <PathFind.h>

int
main ()
{
    Location_t          path[3][5];         // Array of path taken by balls
    int                 lvl;                // Crisscross level
    int                 pl;                 // Player number
    char                buf[512];           // Formatting buffer

    const Location_t    start[] =           // Starting location of balls
        {LOC_LEFT, LOC_MIDDLE, LOC_RIGHT};
    const Location_t        guess[][] = {   // Guesses made by friends
        {LOC_LEFT, LOC_MIDDLE, LOC_RIGHT},
        {LOC_LEFT, LOC_RIGHT, LOC_MIDDLE},
        {LOC_MIDDLE, LOC_LEFT, LOC_RIGHT},
        {LOC_MIDDLE, LOC_RIGHT, LOC_LEFT},
        {LOC_RIGHT, LOC_LEFT, LOC_MIDDLE}
    };

    // Test the function

    FindPath (path, start, guess);

    // Show the result

    strcpy (buf, "R G B");
    cout << buf << '\n';
    for (lvl = 0; lvl < 5; lvl++) {
        buf[path[BALL_RED][lvl] * 2] = 'R';
        buf[path[BALL_GREEN][lvl] * 2] = 'G';
        buf[path[BALL_BLUE][lvl] * 2] = 'B';
        cout << buf << '\n';
    }

    // Verify that no winner is found

    for (pl = 0; pl < 5; pl++) {
        if (
            guess[pl][BALL_RED] == path[BALL_RED][LVL_CNT-1] &&
            guess[pl][BALL_GREEN] == path[BALL_GREEN][LVL_CNT-1] &&
            guess[pl][BALL_BLUE] == path[BALL_BLUE][LVL_CNT-1]
        )
            cout << "** Player " << pl << " wins!\n";
    }
    return 0;
}
```

```
// Java
class Game_c {
    public static void
    main (
        Location_t      args[],             // Starting location of balls
    {
        Location_t      path[][];           // Array of path taken by balls
        int             lvl;                // Crisscross level
        int             pl;                 // Player number
        StringBuilder   str;                // String builder

        final Location_t start[] =          // Starting location of balls
            {Location_t.LOC_LEFT, Location_t.LOC_MIDDLE,
                                                Location_t.LOC_RIGHT};
        final Location_t guess[][] = {      // Guesses made by friends
            {Location_t.LOC_LEFT, Location_t.LOC_MIDDLE,
                                                Location_t.LOC_RIGHT},
            {Location_t.LOC_LEFT, Location_t.LOC_RIGHT,
                                                Location_t.LOC_MIDDLE},
            {Location_t.LOC_MIDDLE, Location_t.LOC_LEFT,
                                                Location_t.LOC_RIGHT},
            {Location_t.LOC_MIDDLE, Location_t.LOC_RIGHT,
                                                Location_t.LOC_LEFT},
            {Location_t.LOC_RIGHT, Location_t.LOC_LEFT,
                                                Location_t.LOC_MIDDLE}
        };

        // Test the function

        path = new Location_t[3][5];
        FindPath (path, start, guess);

        // Show the result

        str = new StringBuilder ("R G B");
        System.out.println (str.toString());
        for (lvl = 0; lvl < 5; lvl++) {
            str.setCharAt(path[BALL_RED][lvl].ordinal() * 2, 'R');
            str.setCharAt(path[BALL_GREEN][lvl].ordinal() * 2, 'G');
            str.setCharAt(path[BALL_BLUE][lvl].ordinal() * 2, 'B');
            System.out.println (str.toString());
        }

        // Verify that no winner is found

        for (pl = 0; pl < 5; pl++) {
            if (
                guess[pl][BALL_RED] == path[BALL_RED][LVL_CNT-1] &&
                guess[pl][BALL_GREEN] == path[BALL_GREEN][LVL_CNT-1] &&
                guess[pl][BALL_BLUE] == path[BALL_BLUE][LVL_CNT-1]
            ) {
                System.out.print ("** Player ");
                System.out.print (pl.toString());
                System.out.println (" wins!");
            }
        }
    }
}
```

```
// C#
using System;
using System.Text;

class Game_c {
    static void
    Main (
        string[]        args)               // Program arguments
    {
        Location_t[][]  path;               // Array of path taken by balls
        int             lvl;                // Crisscross level
        int             pl;                 // Player number
        StringBuilder   str;                // String builder

        Location_t[] start =                // Starting location of balls
            {Location_t.LOC_LEFT, Location_t.LOC_MIDDLE,
                                                Location_t.LOC_RIGHT};
        Location_t[,] guess = {             // Guesses made by friends
            {Location_t.LOC_LEFT, Location_t.LOC_MIDDLE,
                                                Location_t.LOC_RIGHT},
            {Location_t.LOC_LEFT, Location_t.LOC_RIGHT,
                                                Location_t.LOC_MIDDLE},
            {Location_t.LOC_MIDDLE, Location_t.LOC_LEFT,
                                                Location_t.LOC_RIGHT},
            {Location_t.LOC_MIDDLE, Location_t.LOC_RIGHT,
                                                Location_t.LOC_LEFT},
            {Location_t.LOC_RIGHT, Location_t.LOC_LEFT,
                                                Location_t.LOC_MIDDLE}
        };

        // Test the function

        path = new Location_t[3, 5];
        FindPath (path, start, guess);

        // Show the result

        str = new StringBuilder ("R G B");
        Console.WriteLine(str.ToString());
        for (lvl = 0; lvl < 5; lvl++) {
            str[(int)path[BALL_RED, lvl] * 2] = 'R';
            str[(int)path[BALL_GREEN, lvl] * 2] = 'G';
            str[(int)path[BALL_BLUE, lvl] * 2] = 'B';
            Console.WriteLine(str.ToString());
        }

        // Verify that no winner is found

        for (pl = 0; pl < 5; pl++) {
            if (
                guess[pl][BALL_RED] == path[BALL_RED][LVL_CNT-1] &&
                guess[pl][BALL_GREEN] == path[BALL_GREEN][LVL_CNT-1] &&
                guess[pl][BALL_BLUE] == path[BALL_BLUE][LVL_CNT-1]
            )
                Console.WriteLine("** Player " + pl.ToString() + " wins!");
        }
    }
}
```

Gambarajah 3: Contoh Program Ujian

2 KEPUTUSAN CARIAN

Anda baru sahaja menyertai satu syarikat yang baru-baru ini menambah satu bar perintah ke dalam program desktop mereka yang paling popular. Satu tangkapan skrin bar perintah tersebut adalah seperti dalam Gambarajah 4. Bar perintah tersebut menerima perintah teks 6 aksara yang sensitif kepada huruf besar atau kecil sebagai pintasan ke sesebuah skrin.

Gambarajah 4: Tangkapan Skrin untuk Bar Perintah

Bar perintah tersebut juga memaparkan keterangan perintah yang pengguna telah taip. Walau bagaimanapun, memaparkan keterangan untuk perintah semasa tidaklah sangat berguna, khususnya kepada seseorang pengguna yang terlupa perintah tepat yang diperlukannya.

Sebagai seorang pengaturcara yang proaktif, anda mencadangkan supaya bar perintah tersebut harus mencari perintah yang mengandungi teks yang pengguna telah taip dan memaparkannya pada bar tersebut.

Pihak pengurusan bersetuju dengan idea anda tetapi juga mengemukakan keperluan tambahan.

Bar perintah tersebut mesti mencari kedua-dua perintah dan keterangan. Semasa pencarian, bar perintah mesti mengabaikan perbezaan huruf besar / kecil dalam keterangan. Kedua-dua perintah dan keterangan mengandungi aksara ASCII sahaja.

Jika keputusan carian tidak dapat dimuatkan ke bar perintah, bar perintah tersebut mesti mengasingkan keputusan carian ke beberapa halaman.

Untuk setiap perintah, bar perintah tersebut mesti memaparkan perintah tersebut dan keterangannya dalam satu baris pada halaman yang sama. Bar perintah tersebut mesti memaparkan aksara colon ‘:’ di antara perintah dan keterangannya. Bar perintah tersebut mesti memaparkan aksara paip ‘|’ di antara keterangan perintah dan perintah yang seterusnya. Jika sesuatu perintah dan keterangannya tidak dapat dimuatkan ke dalam ruang yang didapati, ia harus ditempatkan pada halaman seterusnya. Walau bagaimanapun, bar perintah tersebut mesti memaparkan sekurang-kurangnya satu pasangan perintah dan keterangan dalam setiap halaman.

Bar perintah tersebut memaparkan teks dalam fon yang sama lebar tetapi tetingkap yang mengandungi bar perintah tersebut boleh diubah saiznya. Oleh sebab itu, bilangan perintah setiap halaman dan bilangan halaman boleh berubah-ubah. Apabila bar perintah tersebut diubah saiznya, ia harus memindah perintah-perintah ke halaman-halaman bergantung kepada kapasiti terkini. Jika nombor halaman yang diminta lebih besar daripada bilangan halaman yang digunakan oleh keputusan carian, bar perintah tersebut harus memaparkan halaman terakhir.

Bar perintah tersebut mesti memaparkan perintah-perintah dalam susunan mengikut keutamaan. Adalah berkemungkinan juga senarai perintah tersebut akan berubah bergantung kepada hak pencapaian pengguna yang sedang log masuk.

Anda perlu memenuhi keperluan-keperluan tersebut dengan terlebih dahulu mengaturcara satu fungsi yang mencari senarai perintah dan menghasilkan keputusan carian. Fungsi anda mesti menerima parameter-parameter berikut:

1. perkataan carian;
2. nombor halaman semasa;
3. satu senarai pasangan nama perintah dan keterangan yang diatur mengikut keutamaan;
4. Kapasiti penimbal yang ditentukan oleh ruang yang didapati pada bar perintah;
5. Penimbal untuk menyimpan keputusan carian.

Gambarajah 5 dan Gambarajah 6 menunjukkan contoh pengisytiharan fungsi tersebut dalam C++.

```
// CommandItem.h
struct CmdItem_t {
    char            cmdItemCmd[6];      // Command
    char            cmdItemDesc[256];   // Description
};
```

Gambarajah 5: Pengisytiharan Struktur Perintah dalam C++

```
// CmdBar.h - Command Bar
class CmdBar_c {
public:

    // Perform search and return result for a particular page

    void CmdBarSearch (
        const char      *word,          // Search word
        int             page,           // Current page number
        const CmdItem_t *itemArr,       // Pointer to items
        size_t          itemCnt,        // Number of items
        size_t          cap,            // Buffer capacity
        char            *result)        // Buffer to store result
    {
        // TODO:
    }
};
```

Gambarajah 6: Pelaksanaan Kelas Bar Perintah dalam C++

3 PENSKALAAN GAMBAR

Sistem operasi Peephole menyimpan gambar-gambar dalam format warna 24-bit yang diterangkan berikut. Peephole mempunyai fungsi-fungsi untuk menangkap dan memaparkan gambar-gambar, tetapi tidak mempunyai keupayaan untuk memotong, mengembang atau mengecut gambar-gambar yang disimpan. Tugas anda ialah mengaturcara satu fungsi yang menerima satu gambar tersimpan dalam format 24-bit Peephole, dan parameter-parameter untuk memotong dan mengembang atau mengecut gambar tersebut, dan kemudiannya menghasilkan satu gambar yang telah dipotong dan dikembang atau dikecut seperti yang diperlukan oleh parameter-parameter tersebut.

Format Peephole 24-bit mengandungi satu kepala diikuti oleh satu tatasusunan struktur warna piksel. Kepala tersebut mempunyai struktur seperti dalam Table 1.

Table 1 – Format Gambar Peephole 24-bit

Offset	Kandungan			
0 ke 3	‘P’	‘H’	‘2’	‘4’
4 ke 7	lebar			
8 ke 11	ketinggian			
12 ke 14	merah[0,0]	hijau[0,0]	biru[0,0]	
15 ke 17	merah[1,0]	hijau[1,0]	biru[1,0]	
: :	:	:	:	
: :	merah[n-1,m-1]	hijau[n-1,m-1]	biru[n-1,m-1]	

4 bait pertama di struktur gambar mengenalpasti struktur tersebut dan mengandungi aksara ‘P’, ‘H’, ‘2’ and ‘4’ dalam turutan tersebut.

Empat bait yang seterusnya mengandungi lebar gambar tersebut dalam piksel dan ini diikuti oleh empat lagi bait yang mengandungi ketinggian gambar tersebut dalam piksel. Kedua-dua lebar dan ketinggian adalah dalam format big endian. Iaitu bait yang pertama mengandungi nilai 8 bit yang paling penting dan bait terakhir mengandungi nilai 8 bit yang paling kurang penting. Sebagai contoh, jika lebar gambar tersebut ialah 6500 piksel, maka empat bait lebar akan mengandungi 0, 0, 25 dan 100 dalam turutan tersebut.

Lebar dan ketinggian tersebut adalah diikuti oleh satu tatasusunan yang berstruktur 3 bait yang mengandungi nilai keamatan merah, hijau dan biru untuk setiap piksel. Piksel-piksel tersebut diatur dengan baris dan kemudian lajur dalam sistem koordinat tangan kanan. Sebagai contoh, jika gambar tersebut mempunyai dua baris dan tiga lajur, baris-baris dan lajur-lajur akan dinomborkan seperti dalam Gambarajah 7 dan disimpan dalam susunan seperti dalam Table 1.

[0,1]	[1,1]	[2,1]
[0,0]	[1,0]	[2,0]

Gambarajah 7 - Cara Menomborkan Piksel

Table 2 – Contoh Susunan Piksel

merah[0,0]	hijau[0,0]	biru[0,0]
merah [1,0]	hijau[[1,0]	biru [1,0]
merah [2,0]	hijau[[2,0]	biru [2,0]
merah [0,1]	hijau[[0,1]	biru [0,1]
merah [1,1]	hijau[[1,1]	biru [1,1]
merah [2,1]	hijau[[2,1]	biru [2,1]

Jika anda mengaturcara dalam C atau C++, fungsi anda mesti mempunyai pengisytiharan seperti berikut:

```
unsigned char *ScaleImage (
     const unsigned char *unscaledImage,
     long                xOfs,
     long                yOfs,
     unsigned long       unscaledWidth,
     unsigned long       unscaledHeight,
     unsigned long       scaledWidth,
     unsigned long       scaledHeight,
     unsigned char       fillRed,
     unsigned char       fillGreen,
     unsigned char       fillBlue);
```

`unscaledImage`
: merupakan penunjuk ke satu tatasusunan bait yang mengandungi gambar yang belum diskalakan dalam format Peephole 24-bit.

`xOfs`
: merupakan offset melintang dari sudut kiri dari bawah bagi gambar yang belum diskala tersebut sehingga ke pinggir kiri segi empat pemotongan dalam piksel. `xOfs` mungkin bernilai negatif.

`yOfs`
: merupakan offset menegak dari sudut kiri dari bawah bagi gambar yang belum diskala tersebut sehingga ke pinggir bawah segi empat pemotongan dalam piksel. `yOfs` mungkin bernilai negatif.

`unscaledWidth`
: merupakan lebar segi empat pemotongan dalam piksel.

`unscaledHeight`
: merupakan ketinggian segi empat pemotongan dalam piksel.

`scaledWidth`
: merupakan lebar gambar yang diskala dalam piksel.

`scaledHeight`
: merupakan ketinggian gambar yang diskala dalam piksel.

`fillRed`
: merupakan komponen merah untuk isi warna (0 ke 255).

`fillGreen`
: merupakan komponen hijau untuk isi warna (0 ke 255).

`fillBlue`
: merupakan komponen biru untuk isi warna (0 ke 255).

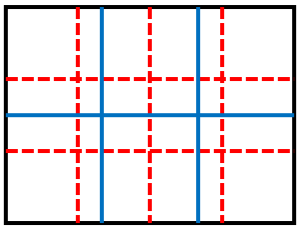
`Return value`
: merupakan satu penunjuk kepada satu tatasusunan bait dalam ingatan dinamik (iaitu ingatan yang diperuntukkan dengan sama ada **new** operator atau fungsi **malloc**) yang mengandungi gambar yang telah diskalakan tersebut.

`ScaleImage` mesti meregang atau memampat segi empat pemotongan tersebut yang diwakili oleh parameter-parameter `xOfs`, `yOfs`, `unscaledWidth` dan `unscaledHeight` ke dalam satu gambar baru yang mempunyai dimensi `scaledWidth` dan `scaledHeight` dan kemudiannya memulangkan satu penunjuk ke gambar baru tersebut, di mana ia mesti disimpan dalam ingatan yang diperuntukkan secara dinamik.

Segi empat pemotongan tersebut mungkin tidak berada sepenuhnya dalam sempadan gambar yang belum diskala tersebut. Dalam keadaan ini, `ScaleImage` mesti memenuhkan isi warna seperti ditakrifkan oleh parameter-parameter `fillRed`, `fillGreen` dan `fillBlue`, ke dalam piksel-piksel dalam segi empat pemotongan yang berada di luar sempadan gambar yang belum diskala itu.

`ScaleImage` mesti memenuhi setiap piksel dalam gambar terskala dengan warna purata di kawasan penindihan dalam segi empat pemotongan yang mana piksel dalam gambar terskala bertindih.

Sebagai contoh, katakan satu segi empat pemotongan 4 x 3 (dalam garis putus-putus merah) dan satu gambar terskala 3 x 2 (dalam garis biru) dalam Gambarajah 8



Gambarajah 8 – Segi Empat Pemotongan 4 x 3 dan Gambar Terskala 3 x 2

Dalam kes ini, piksel [1,0] dalam gambar berskala bertindih dengan piksel-piksel [1,0], [2,0], [1,1] dan [2,1] bagi segi empat pemotongan dan kawasan bertindih adalah 2/6, 2/6, 1/6 dan 1/6 masing-masing, maka keamatan merah untuk piksel [1,0] dalam gambar terskala ialah:

```
(2*red[1,0] + 2*red[2,0] + red[1,1] + red[2,1]) / 6
```

Apabila mengira keamatan piksel, `ScaleImage` mesti membundarkan nombor yang diberi oleh warna purata ke integer yang terdekat. Contohnya jika keamatan merah yang diberi oleh formula di atas ialah 155/6, `ScaleImage` mesti membundarkan keamatan ke 26 dan jika keamatan diberi oleh formula di atas ialah 151/6, ia mesti membundarkan keamatan ke 25. Jika keamatan teori ialah di tengah-tengan antara dua integer, contohnya 153/6, `ScaleImage` boleh membundarkan keamatan teori tersebut sama ada ke atas atau ke bawah sehingga ke integer terdekat.

Aritmetik titik terapung di bawah sistem operasi Peephole adalah tidak laju ataupun tepat. `ScaleImage` tidak harus menggunakan aritmetik titik terapung.

Seperti dinyatakan sebelum ini, sistem operasi Peephole tidak menyokong pemotongan atau pengskalaan gambar, maka fungsi-fungsi yang berdasarkan sebarang bahasa pengaturcaraan yang biasanya didapati untuk pengubahsuaian gambar adalah tidak wujud dalam implementasi Peephole bagi bahasa tersebut dan `ScaleImage` mestilah tidak menggunakan fungsi-fungsi tersebut.

4 LAPORAN PESANAN YANG TIDAK DIKESAHKAN

Satu syarikat bernama Order Me Around (OMA) mengendalikan satu sistem dalam talian untuk pesanan makanan. Sistem tersebut menerima pesanan makanan dari aplikasi mudah alih dan pelayar web yang digunakan oleh pengguna-pengguna yang berada di seluruh negara.

Apabila sistem tersebut menerima satu pesanan, ia memilih restoran-restoran di antara yang terbaik berdasarkan kedekatan kepada pengguna, bilangan pesanan di restoran dan penilaian pengguna untuk restoran-restoran tersebut. Sistem tersebut akan menyampaikan pesanan tersebut kepada restoran-restoran terpilih tersebut. Restoran-restoran boleh pilih untuk menerima atau menolak pesanan tersebut. Restoran pertama yang pilih untuk menerima pesanan akan mendapat kerja untuk mengendalikan pesanan tersebut dan menghantar makanan tersebut. Sistem tersebut membenarkan restoran untuk menukar keputusan mereka. Jika restoran yang mengendalikan kerja tersebut menolak pesanan tersebut, sistem akan menugaskan pesanan tersebut kepada restoran seterusnya yang pilih untuk menerima pesanan tersebut.

Kadang-kala, pelanggan mengadu bahawa mereka tidak menerima pesanan mereka dan meminta bayaran balik. Tidak ada sebarang cara untuk membuktikan bahawa makanan tersebut telah dihantar atau tidak. Dalam kebanyakan kes, OMA tiada pilihan dan terpaksa bayar balik kepada pelanggan dan juga bayar restoran untuk perkhidmatan tersebut, untuk mengekalkan hubungan baik dengan pelanggan dan dengan restoran-restoran.

Untuk mengatasi masalah ini dan mengelakkan kerugian yang selanjutnya, OMA mengambil keputusan untuk melaksanakan satu prosedur pengesahan. Apabila sesuatu pesanan dihantar, sistem akan memberi pengguna tersebut satu kod rahsia. Pengguna tersebut mesti memberi kod rahsia tersebut kepada orang yang menghantar supaya menerima makanannya. OMA akan membekalkan penghantar satu aplikasi mudah alih baru yang akan mengesahkan kesahihan kod rahsia tersebut. Jika kod rahsia tidak disahkan, OMA tidak akan melepaskan bayaran dari pelanggan kepada restoran.

Sebagai sebahagian daripada pasukan pembangunan, kerja anda ialah mengaturcara satu program yang menghasilkan satu laporan yang mengandungi senarai transaksi di mana kod rahsianya tidak dikesahkan. Laporan tersebut mesti memaparkan maklumat berikut dalam aturan seperti ditunjukkan dalam Gambarajah 9:

1. untuk setiap restoran (disusun mengikut abjad):
    a. nama restoran;
    b. untuk setiap pesanan yang dipenuhi tetapi tidak dikesahkan (disusun mengikut nombor pesanan):
        i. nombor pesanan;
        ii. nilai pesanan;
    c. jumlah nilai pesanan bagi pesanan yang dipenuhi tetapi tidak dikesahkan;
    d. bilangan pesanan yang dikesahkan;
    e. peratusan pesanan yang dikesah atas pesanan yang dipenuhi (dibulatkan atau dicicirkan ke 2 tempat perpuluhan);
2. jumlah besar nilai pesanan bagi pesanan yang dipenuhi tetapi tidak dikesahkan;
3. peratusan jumlah pesanan dikesah atas jumlah pesanan dipenuhi (dibulatkan atau dicicirkan ke 2 tempat perpuluhan).

Sistem tersebut mempunyai jadual pangkalan data seperti dalam Gambarajah 10. Masa adalah disimpan dalam unit saat sejak tengah malam pada 1hb Januari 1970. Nilai disimpan dalam integer sen. Sebagai contoh, nilai 1.0 ialah 1 sen, dan 100.0 ialah RM 1.

```
======== UNVERIFIED ORDERS REPORT ========

RESTAURANT           ORDER NO      VALUE VERIFIED   (%)
-------------------- -------- ---------- ------ -------
KL-FG-CD               334567     120.40
                       334656     124.95
                              ----------
TOTAL                             245.35      8   80.00
                              ----------

MR DUNELD'S            334455     320.00
                       334456      24.95
                              ----------
TOTAL                             344.95     18   95.00
                              ----------

POPEARS                                      10  100.00

-------------------- -------- ---------- ------ -------
GRAND TOTAL                       590.30     36   90.00
-------------------- -------- ---------- ------ -------
```

Gambarajah 9: Contoh Laporan

```
// Customer order

exec sql create table ord (
    ordOrderNo      integer not null,           // Order number
    ordTim          double precision not null,  // Order time
    ordValue        integer not null,           // Order value
    ordVerified     smallint not null,          // Verified flag (0=N,1=Y)
);
exec sql create unique cluster index ordOrderNoInd on
    ord (ordOrderNo);
```

```
// Order response from restaurants

exec sql create table ordRsp (
    ordRspOrderNo   integer not null,           // Order number
    ordRspRestId    integer not null,           // Restaurant id
    ordRspTim       double precision,           // Response time
    ordRspAccept    smallint                    // Accept flag (0=N,1=Y)
);
exec sql create index ordRspOrderNoInd on
    ordRsp (ordRspOrderNo, ordRspRestId, ordRspTim);
```

```
// Restaurant record

exec sql create table res (
    resRestId       integer not null,           // Restaurant id
    resRestName     char(21) not null           // Restaurant name
);
exec sql create unique cluster index resRestIdInd on
    res (resRestId);
```

Gambarajah 10: Skema Pangkalan Data

Table 3 menunjukkan sampel rekod dalam jadual balasan pesanan. Dalam data sampel tersebut, restoran 4 memenuhi pesanan 121277 kerana restoran 2 menolak pesanan tersebut setelah menjadi yang pertama untuk menerima pesanan tersebut. Restoran 6 memenuhi pesanan 121278, sementara restoran 4 memenuhi pesanan 121279.

Order Number	Restaurant ID	Response Time	Accept Flag
121277	2	1510484400	1
121278	6	1510484410	1
121277	2	1510484445	0
121279	4	1510484447	1
121277	4	1510484450	1

Table 3: Sampel Data dalam Jadual Balasan Pesanan