# Collegiate Programming Examination(CPE) Manual

2018/5/29

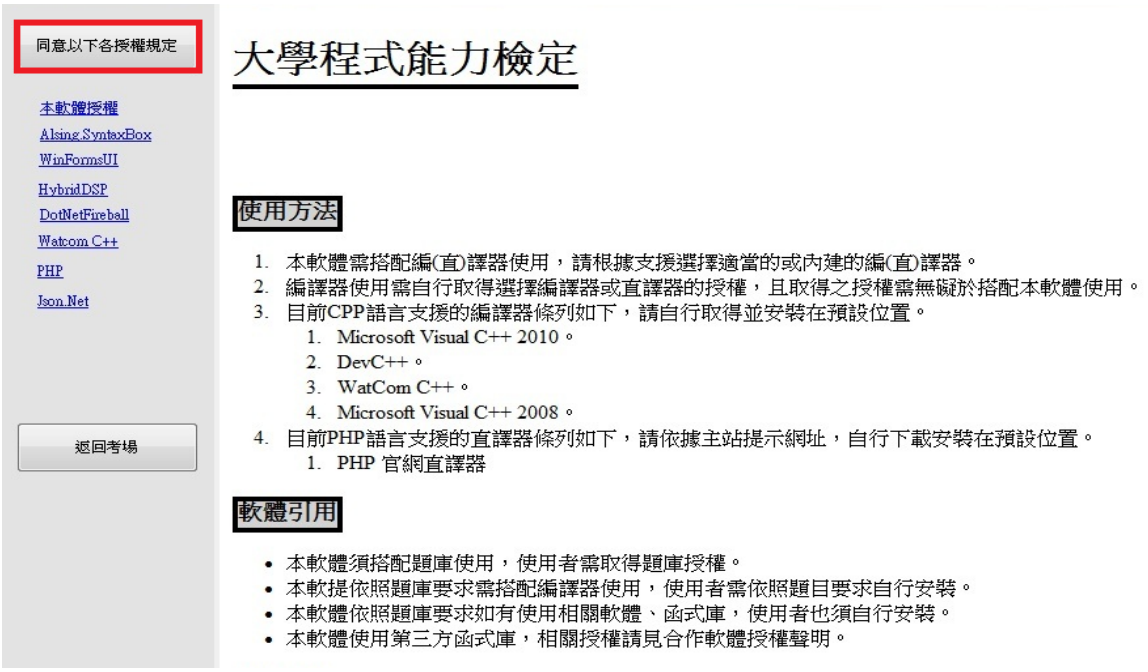
This handbook is provided to examinees who take the CPE exam. The content includes the use of the exam environment and the basics of the I/O used when solving problems.

1. To take the CPE exam, please complete the account registration on the CPE website before signing up for the exam. If you already have an account, you can just sign up for the exam without registration. Please carefully select the examination room when you sign up for the exam and go to report for your identity and take the exam at the designated time.
2. From 2013/10/1, CPE exam adopts new evaluation system called Coding Frenzy: http://coding-frenzy.arping.me/
   Examinees can go to the "Coding Frenzy" to register and practice online. (Note: CPE account is different from Coding Frenzy account. CPE account must be used for CPE exam and Coding Frenzy account must be used for Coding Frenzy exercise.)
3. The CPE website, http://cpe.cse.nsysu.edu.tw/, has more detailed information on "環境與教材".

# 1. CPE examination environment

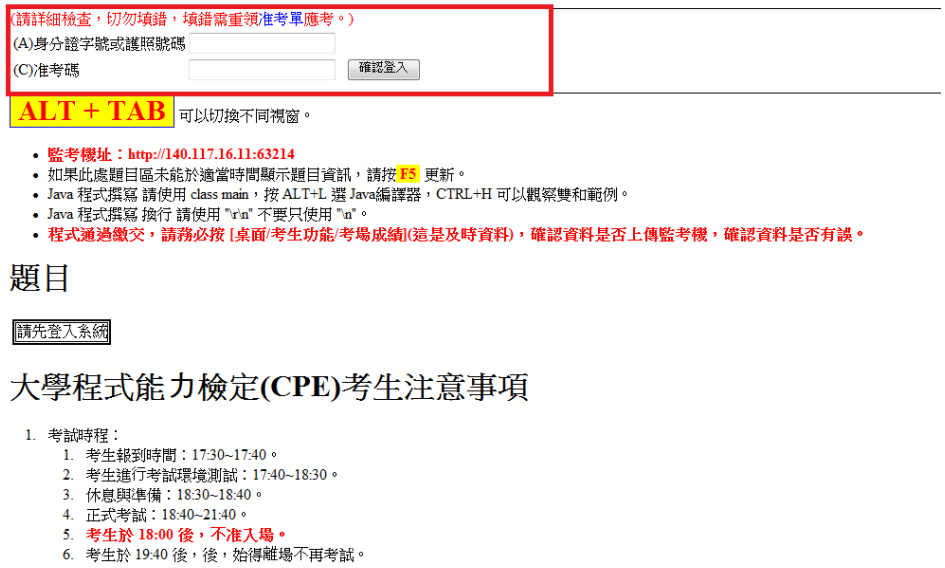
## 1.1 Coding Frenzy: System Registration and Login

Please click "同意以下各授權規定" to enter the login screen of the Coding Frenzy system, as shown in Figure 1.1.1.。




**(Figure 1.1.1)**

After entering, please complete the "identity card" and "test number" (the test number will be given by the invigilator before the exam) to complete the login action, as shown in Figure 1.1.2




**(Figure1.1.2)**

After completing the login screen as shown in Figure 1.1.3, examinees can see the menu, the current time, and the list of exam topics.


**(Figure1.1.3)**

## 1.2 Menu

Including Clarification Request, Clarification Response, Score board, Dictionary, and some additional tools, such as Figure 1.2.1.


**(Figure1.2.1)**

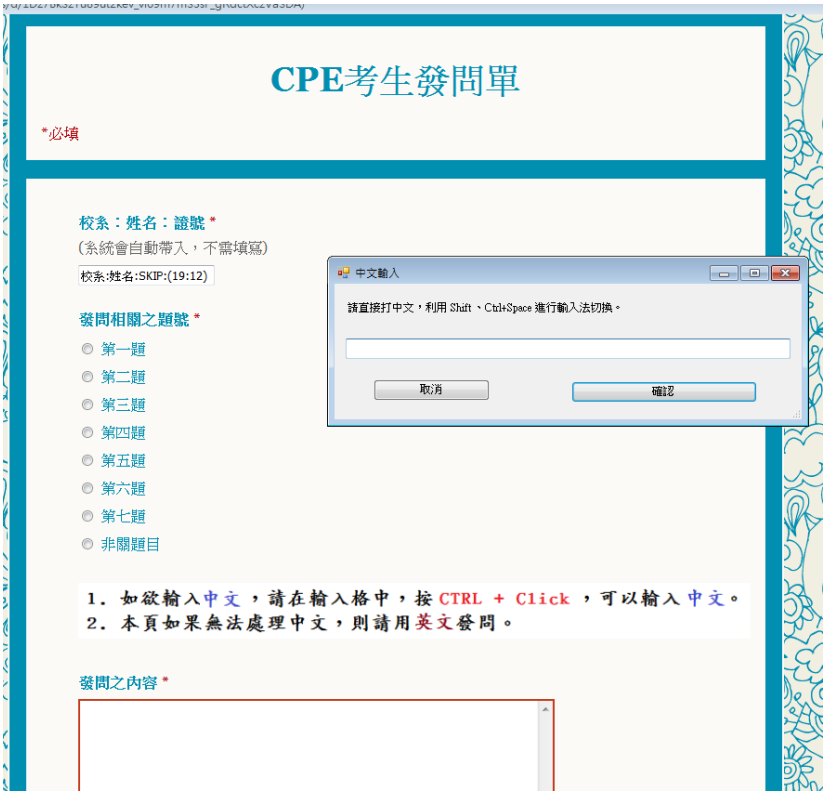
### 1.2.1 **Clarification Request**

After clicking, there is a request form. Please fill in the questions you want to ask

and related information (question number, description of the problem), as shown in Figure 1.2.2. After the question is sent out, only the questions which are answered by the invigilator will be displayed, as shown in Figure 1.2.3.

- **Chinese Input Method**: In the exam room, shortcut keys should already be set Alt-Left + Shift +1、Alt-Left + Shift + 2、Alt-Left + Shift +3 to switch input methods.

- **Chinese Input Method**: If the shortcut keys are not set, you need to input through additional windows, hold down the “CTRL” key and left click on the field you want to enter, and use the “new phonetic” input in the resulting window.
  On the examination room




**(Figure1.2.2)**

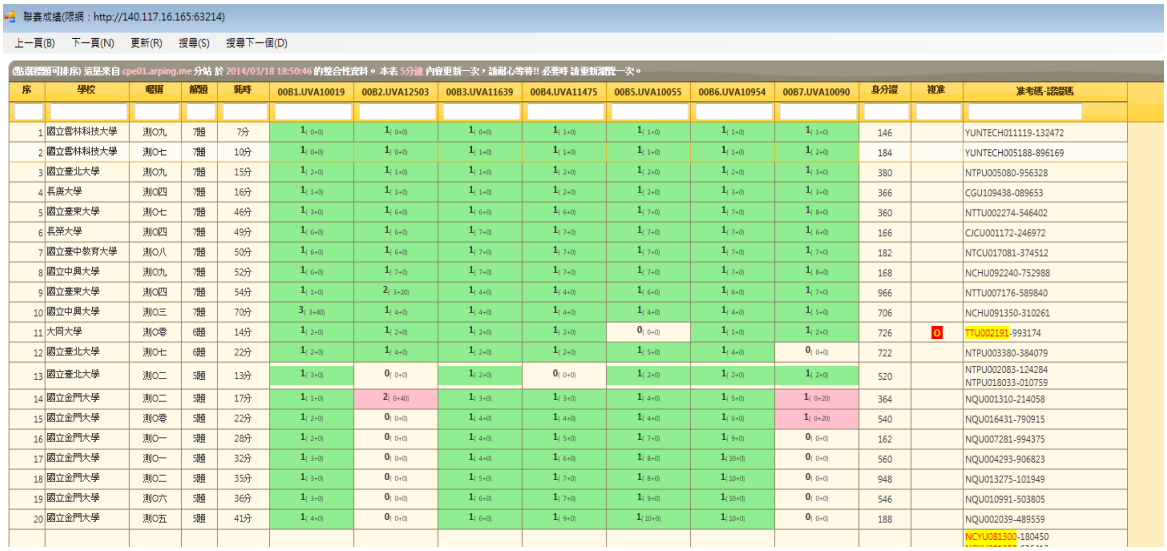

**(Figure1.2.3)**

### 1.2.2 **Score board(practice)**

The current results of the candidates for the CPE exercise (number of questions and the time for solving the questions) are shown. The range includes all candidates for the examination room. This grade is updated every five minutes.

### 1.2.3 **Score board**

Show the real-time results (number of questions and the time for solving the questions) of examinees for this CPE exam, as shown in Figure 1.2.4. The scope includes all examinees. This grade is updated every five minutes.




| 序 | 學校 | 暱稱 | 題題 | 耗時 | 00B1.UVA10019 | 00B2.UVA12503 | 00B3.UVA11639 | 00B4.UVA11475 | 00B5.UVA10055 | 00B6.UVA10954 | 00B7.UVA10090 | 身分證 | 複查 | 准考證-認證碼 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 國立雷林科技大學 | 測O九 | 7題 | 7分 | 1( 0+0) | 1( 0+0) | 1( 0+0) | 1( 1+0) | 1( 1+0) | 1( 1+0) | 1( 1+0) | 146 | | YUNTECH011119-132472 |
| 2 | 國立雷林科技大學 | 測O七 | 7題 | 10分 | 1( 0+0) | 1( 0+0) | 1( 1+0) | 1( 1+0) | 1( 1+0) | 1( 1+0) | 1( 2+0) | 184 | | YUNTECH005188-896169 |
| 3 | 國立臺北大學 | 測O九 | 7題 | 15分 | 1( 2+0) | 1( 1+0) | 1( 1+0) | 1( 2+0) | 1( 2+0) | 1( 2+0) | 1( 3+0) | 380 | | NTPU005080-956328 |
| 4 | 長庚大學 | 測O四 | 7題 | 16分 | 1( 1+0) | 1( 1+0) | 1( 1+0) | 1( 2+0) | 1( 2+0) | 1( 3+0) | 1( 3+0) | 366 | | CGU109438-089653 |
| 5 | 國立臺東大學 | 測O七 | 7題 | 46分 | 1( 3+0) | 1( 6+0) | 1( 6+0) | 1( 6+0) | 1( 7+0) | 1( 7+0) | 1( 8+0) | 360 | | NTTU002274-546402 |
| 6 | 長榮大學 | 測O四 | 7題 | 49分 | 1( 6+0) | 1( 6+0) | 1( 7+0) | 1( 7+0) | 1( 7+0) | 1( 7+0) | 1( 6+0) | 166 | | CJCU001172-246972 |
| 7 | 國立臺中教育大學 | 測O八 | 7題 | 50分 | 1( 6+0) | 1( 6+0) | 1( 7+0) | 1( 7+0) | 1( 7+0) | 1( 7+0) | 1( 7+0) | 182 | | NTCU017081-374512 |
| 8 | 國立中興大學 | 測O九 | 7題 | 52分 | 1( 6+0) | 1( 7+0) | 1( 7+0) | 1( 7+0) | 1( 7+0) | 1( 7+0) | 1( 8+0) | 168 | | NCHU092240-752988 |
| 9 | 國立臺東大學 | 測O四 | 7題 | 54分 | 1( 1+0) | 2( 1+20) | 1( 4+0) | 1( 4+0) | 1( 6+0) | 1( 6+0) | 1( 7+0) | 966 | | NTTU007176-589840 |
| 10 | 國立中興大學 | 測O三 | 7題 | 70分 | 3( 3+40) | 1( 4+0) | 1( 4+0) | 1( 4+0) | 1( 4+0) | 1( 4+0) | 1( 5+0) | 706 | | NCHU091350-310261 |
| 11 | 大同大學 | 測O零 | 6題 | 14分 | 1( 2+0) | 1( 2+0) | 1( 2+0) | 1( 2+0) | 0( 0+0) | 1( 1+0) | 1( 2+0) | 726 | Q | TTU002191-993174 |
| 12 | 國立臺北大學 | 測O七 | 6題 | 22分 | 1( 2+0) | 1( 4+0) | 1( 2+0) | 1( 2+0) | 1( 5+0) | 1( 4+0) | 0( 0+0) | 722 | | NTPU003380-384079 |
| 13 | 國立臺北大學 | 測O二 | 5題 | 13分 | 1( 3+0) | 0( 0+0) | 1( 2+0) | 0( 0+0) | 1( 2+0) | 1( 2+0) | 1( 2+0) | 520 | | NTPU002083-124284<br>NTPU018033-010759 |
| 14 | 國立金門大學 | 測O二 | 5題 | 17分 | 1( 1+0) | 2( 0+40) | 1( 3+0) | 1( 3+0) | 1( 4+0) | 1( 5+0) | 1( 0+20) | 364 | | NQU001310-214058 |
| 15 | 國立金門大學 | 測O零 | 5題 | 22分 | 1( 2+0) | 0( 0+0) | 1( 4+0) | 1( 4+0) | 1( 4+0) | 1( 6+0) | 1( 0+20) | 540 | | NQU016431-790915 |
| 16 | 國立金門大學 | 測O一 | 5題 | 28分 | 1( 2+0) | 0( 0+0) | 1( 4+0) | 1( 5+0) | 1( 7+0) | 1( 9+0) | 0( 0+0) | 162 | | NQU007281-994375 |
| 17 | 國立金門大學 | 測O一 | 5題 | 32分 | 1( 3+0) | 0( 0+0) | 1( 4+0) | 1( 6+0) | 1( 8+0) | 1( 10+0) | 0( 0+0) | 560 | | NQU004293-906823 |
| 18 | 國立金門大學 | 測O二 | 5題 | 35分 | 1( 3+0) | 0( 0+0) | 1( 5+0) | 1( 7+0) | 1( 8+0) | 1( 10+0) | 0( 0+0) | 948 | | NQU013275-101949 |
| 19 | 國立金門大學 | 測O六 | 5題 | 36分 | 1( 3+0) | 0( 0+0) | 1( 6+0) | 1( 7+0) | 1( 9+0) | 1( 10+0) | 0( 0+0) | 546 | | NQU010991-503805 |
| 20 | 國立金門大學 | 測O五 | 5題 | 41分 | 1( 4+0) | 0( 0+0) | 1( 6+0) | 1( 9+0) | 1( 10+0) | 1( 10+0) | 0( 0+0) | 188 | | NQU002039-489559 |
| | | | | | | | | | | | | | | NCYU081300-180450 |

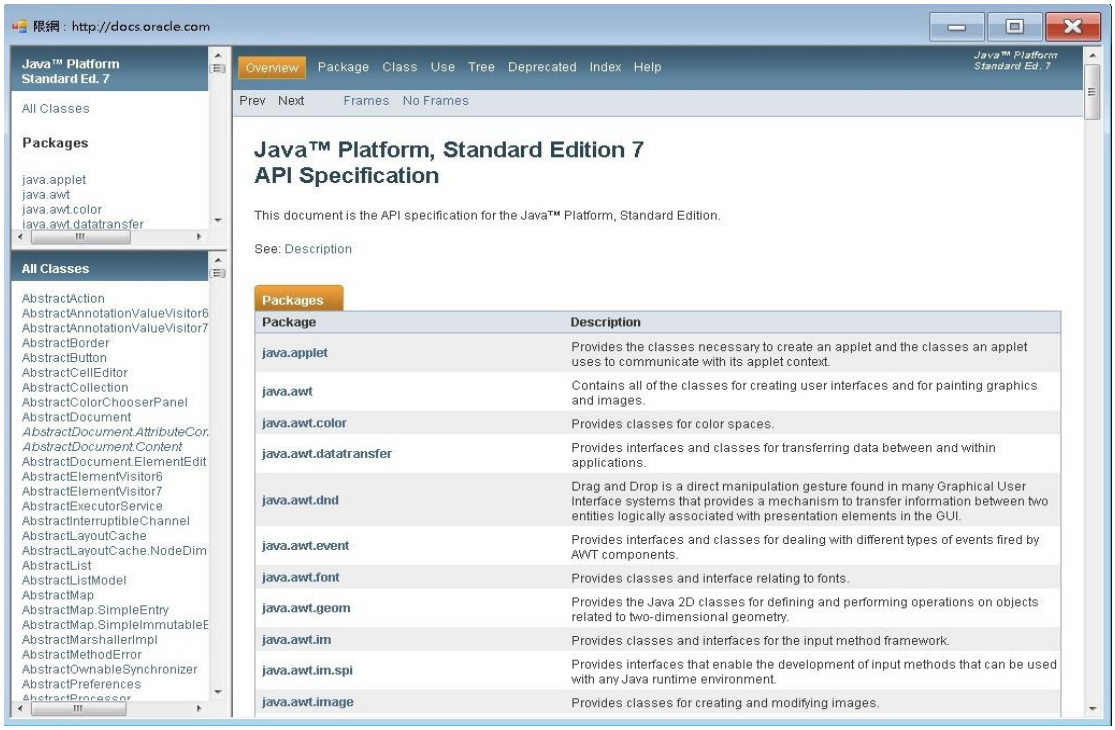
**(Figure1.2.4)**

### 1.2.4 **Score board(examination room)**

Including only the current results of the candidate's current examination room, this result is updated at any time.

### 1.2.5 **Java Reference**

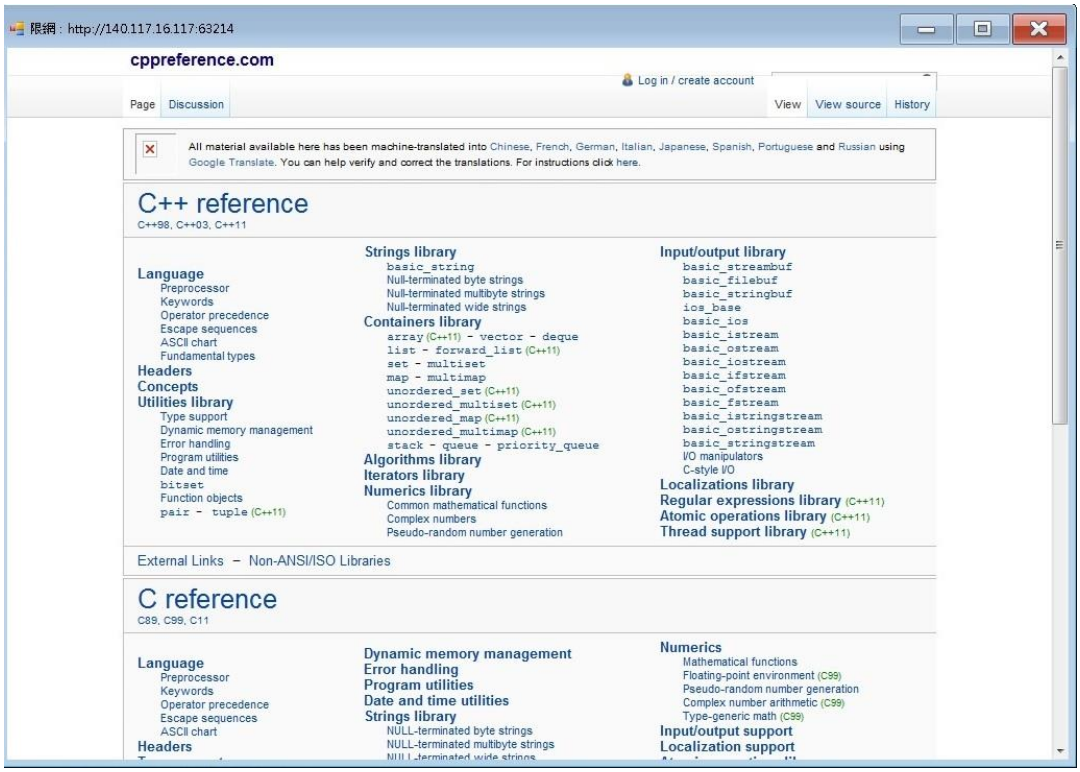
You can query Java syntax, data type, and class member functions, as shown in Figure 1.2.5. Source：http://docs.oracle.com/javase/7/docs/api/




**(Figure1.2.5)**

### 1.2.6 C++ Reference

You can query C++ syntax, data type, and class member functions, as shown in Figure1.2.6。Source：http://en.cppreference.com/w/




**(Figure1.2.6)**

### 1.2.7 **Dictionary**

You can query the meaning of English words, but the English words in the problem cannot be directly copied and pasted.

### 1.2.8 **Time left**

The remaining time of the exam will be displayed in the upper right corner of the window, as shown in Figure 1.2.7.

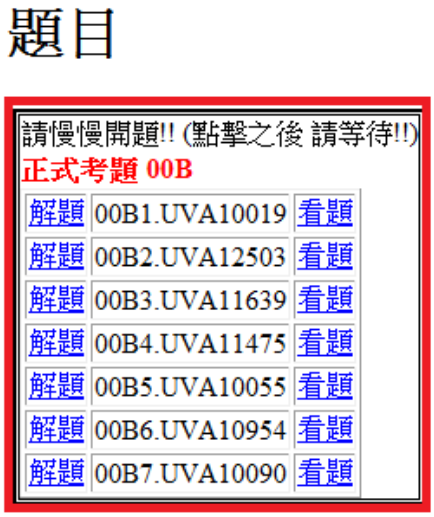

(Figure1.2.7)

### 1.2.9 **End the evaluation**

End this exam. If you do not press it carefully, please report to the invigilator.

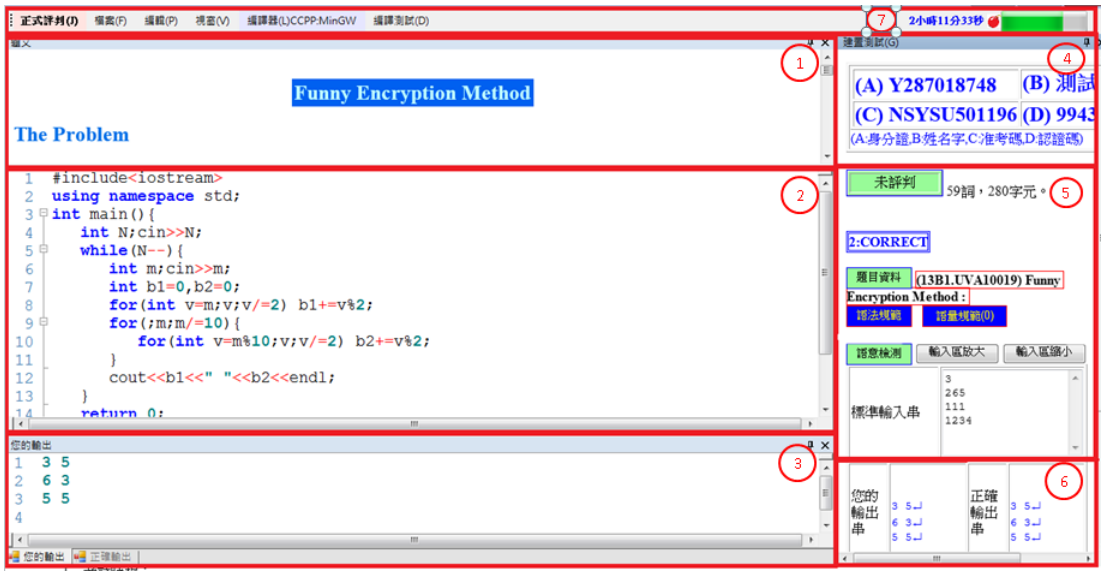
## 1.3 Exam topics

As shown in Figure 1.3.1, click on “解題” below the “正式考題” to enter the problem solving window. If there are no topics, please refresh the page again.




**(Figure1.3.1)**

## 1.4 Exam interface




**(Figure1.4.1)**

After entering the topics, the window is shown in Figure 1.4.1, and the functions of each part are as follows:

(1) Introduction and description of topics.

(2) Program-writing window (Please write your program here).

(3) The output of your program and the correct output of using measurement data.。

(4) The examinee's ID, name, and test number are displayed.
(5) Topic data, grammar specification, language specification, semantic detection. Grammar specification shows compilation error information. The language specification is the syntax that is not allowed in the display code. Semantic detection requires the input of test data.
(6) According to the input data of the semantic detection and the output data generated by the examinee's code, the left half is the output of the examinee's code and the right half is the correct output of the topic.
(7) Function Options, and Remaining Time.

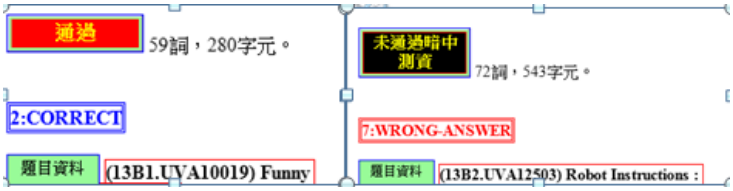
The functions of the function options are described below.

### 1.4.1 Formal judgement (J)

This is the "formal judgement" of the exam. The results of the judgement will be included in the official record of the exam and will also be displayed on the score sheets of all candidates. If the result of the judgement is not passed, penalty points will be applied according to the scoring rules. Examinees should perform "formal judgement" after conducting various tests. After clicking "正式評判(J)", a confirmation window will appear. If you click (Y) in the confirmation window, you can "formal judge" the examinee's program, as shown in Figure 1.4.2.


**(Figure1.4.2)**

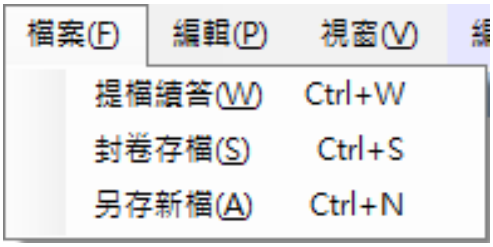
After a formal judgement, if the examinee passes the "Secret Test Data" (unlisted) of this question, the system will show "通過(pass)", otherwise, it will display "未通過暗中測資(not pass)", as shown in Figure 1.4.3.




**(Figure1.4.3)**

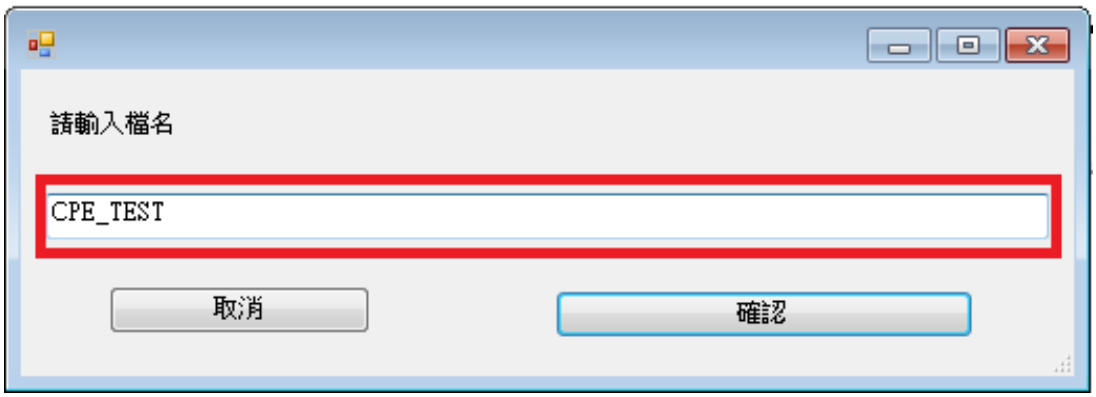
### 1.4.2 File (F)

As shown in Figure 1.4.4, the function of this option is to load file (Ctrl+W), save file (Ctrl+S), and save a new file (Ctrl+N).



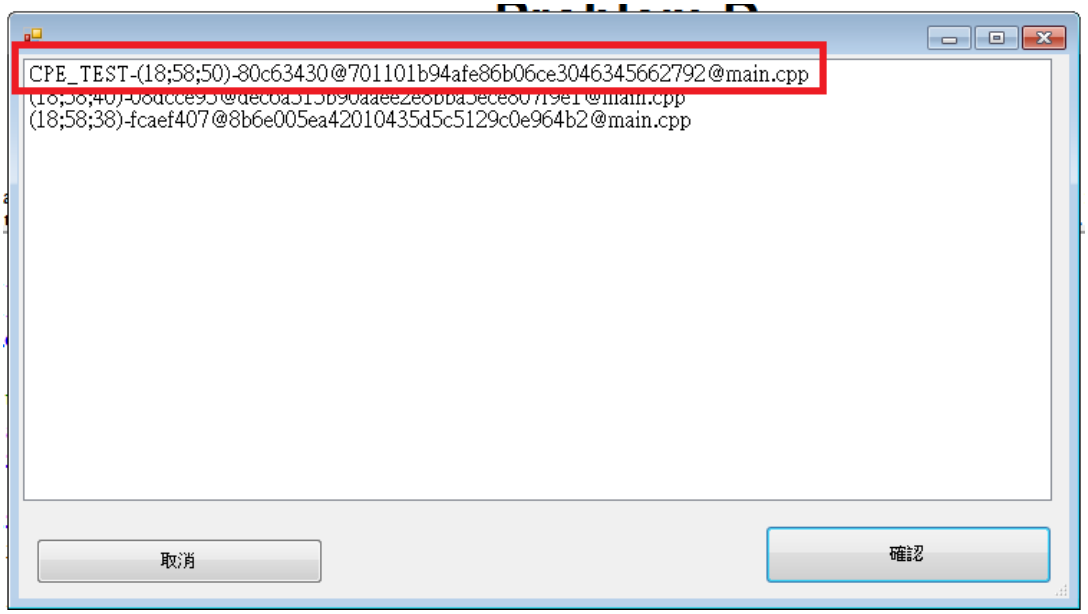

**(Figure1.4.4)**

**Save a new file (Ctrl+N)**: The current code can be saved according to the file name which is input by the examinee, as shown in **Figure 1.4.5**.



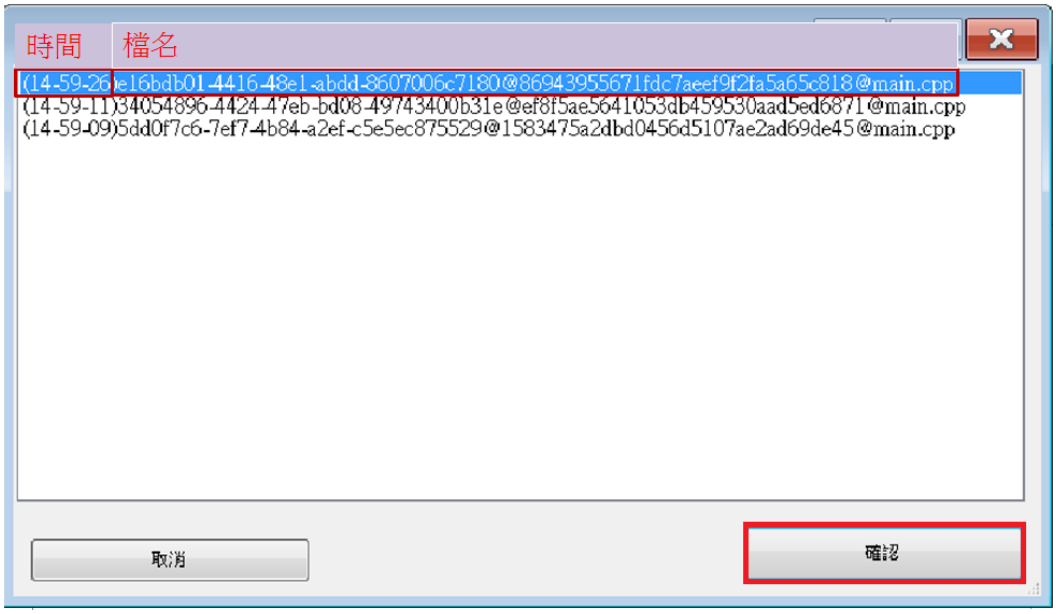

**(Figure 1.4.5)**

**Load file**: After load the saved file (for example, CPE_TEST), you can continue writing the program, as shown in **Figure 1.4.6**.




**(Figure 1.4.6)**

**Save file**: Save current code in chronological order. The file name is randomly generated by the system. After that, you can load the file you had saved, as shown in **Figure 1.4.7**.




**(Figure 1.4.7)**

### 1.4.3 Edit (P)

This function can enlarge or reduce the font size of the program, or it can attach the sample code to examinee for reference. The sample code will be generated in the form of comments at the bottom of the edit window, such as Figure 1.4.8.


**(Figure 1.4.8)**

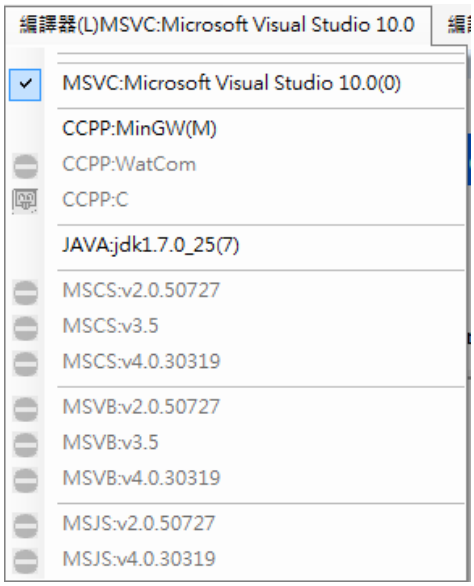
### 1.4.4 View (V)

According to your preferences, you can close the current information window, such as topics, build tests, correct output, and your output. To restore the associated window, just click it again, as shown in **Figure 1.4.9**.


**(Figure 1.4.9)**

### 1.4.5 Compiler (L)

Examinees can choose the language compiler they are familiar with to compile the code in this option, but the prerequisite must be the language allowed by the topic, as shown in **Figure 1.4.10**.


編譯器(L)MSVC:Microsoft Visual Studio 10.0
MSVC:Microsoft Visual Studio 10.0(0)
CCPP:MinGW(M)
CCPP:WatCom
CCPP:C
JAVA:jdk1.7.0_25(7)
MSCS:v2.0.50727
MSCS:v3.5
MSCS:v4.0.30319
MSVB:v2.0.50727
MSVB:v3.5
MSVB:v4.0.30319
MSJS:v2.0.50727
MSJS:v4.0.30319

**(Figure 1.4.10)**

### 1.4.6 Compile and test


**(Figure 1.4.11)**

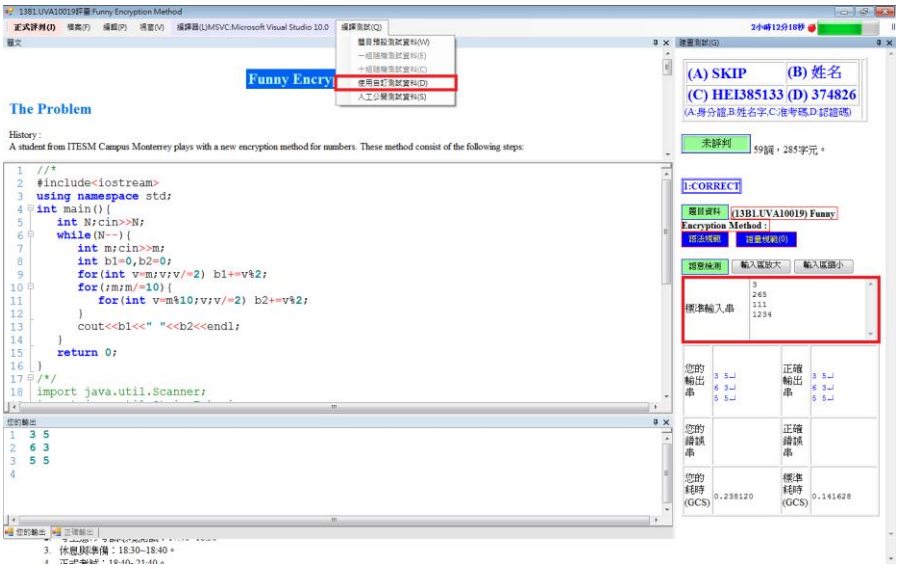
The compile test is to examine the functions of the candidate's code, as shown in **Figure 1.4.11**. The functions are described as follows:

(1) Topic preset test data：

Candidates can use this function to execute and validate the code, and the test data is generated by the system (less difficulty). This test is not included in the test score record.

(2) Use custom test data：

Examinees can use this function to execute and validate the code. The test data is entered by yourself as shown in **Figure 1.4.12**. This test is not included in the test score record.



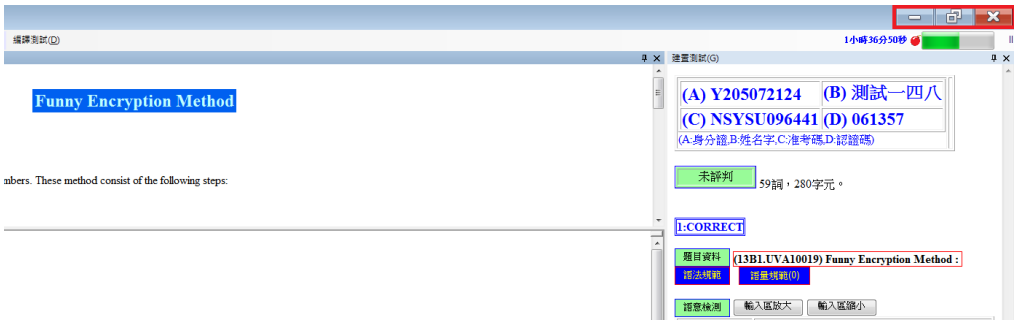

**(Figure 1.4.12)**

(3) Manually open test data：

Examinees can use this function to execute and verify the code. The test data is selected by the proposition teacher according to the questions. The difficulty level is about the same as the "hidden test data" in the system.

### 1.4.7 Change the topic

Candidates who want to return to the initial window to select a topic can narrow or close all the topic windows to return to the selection window, as shown in **Figure 1.4.1**.




**(Figure 1.4.13)**

## 1.5 Result of testing data

After testing or formal judgement, the system will display relevant information as follows:

| | |
|---|---|
| COMPILER-ERROR | The code did not compile successfully. (Click on the link to check the error message generated by the compiler) |
| CORRECT | The program is correct and passes the test. |
| NO-OUTPUT | The program did not output any data. |
| PENDING | The code is still in progress. |
| PRESENTATION-ERROR | The output is correct, but the format is incorrect, such as not following a specified space or line feed (multiple spaces or fewer spaces, multiple line feeds, or fewer line feeds). |
| RUN-ERROR | The execution of the program could not be completed successfully. That is, an error occurred during program execution, such as a memory access error. |
| TIMELIMIT | The execution time of the program exceeds the problem limit. The program may fall into endless loops, or it must need improving the solution method. |
| WRONG-ANSWER | The result of the output is wrong. (This may also be caused by an error in the output format.) |

# 2. C/C++ basic input and output

In the CPE exam's programming, all inputs and outputs take the standard input (stdin) and the standard output (stdout). Reading and writing file are not allowed. When writing programs, you can use functions such as scanf and printf in the C language; in C++, you can use objects such as cin and cout. The input and output data are all pure text data. The program must be written according to the input and output formats of the questions. Moreover, the format of "hidden test data" of is guaranteed to follow the format described by the question, so examinees do not have to check the correctness of the input data.

It would be a pity if candidates don't know how to read the data and further tried to write the functions required by the question. Here will introduce some common types of test data, reading methods, and attach examples of actual questions, hoping to lead you quickly to the problem-solving situation. After each sub-section, there are related exercises. Try to do some exercises that will definitely help the CPE test scores.

## 2.1 C scanf and printf

When it comes to input and output, there are two functions scanf and printf in the C language. Among them, the writing format of scanf is as follows:

```
int scanf(const char *format, ...);
```

The first parameter of the function is a string where you describe how to read the next input. For the format and type of data that you intend to read, fill in the corresponding symbols, as shown in the following table:

| Character | char* | %c | String | char* | %s |
|---|---|---|---|---|---|
| | | int | Unsigned int | Octal | Hexadecimal |
| (unsigned) char* | | %hhd | %hhu | %hho | %hhx |
| (unsigned) short* | | %hd | %hu | %ho | %hx |
| (unsigned) int* | | %d | %u | %o | %x |
| (unsigned) long* | | %ld | %lu | %lo | %lx |
| (unsigned) long long* | | %lld | %llu | %llo | %llx |
| Floating point | float* | %f | float | double* | %lf |

The next parameter is the address of the corresponding data variable, which is preceded by the & symbol. Note that the string variable name itself represents an

address, so only the string does not need to be followed by an ampersand.

For example, the following program:

```
char ch,str[64];
int num;
float value;

scanf("%c%s%d%f",&ch,str,&num,&value);
```

Excepting for %c, all input methods ignore the extra whitespace in front of them, such as space, tab, and enter. Therefore, it is actually indifferent no matter how many the space or the gap is if the sequence of the data is the same. For example, the following conditions can be correctly read as scanf("%d%f%f%f", ...).

| | | | |
|---|---|---|---|
| 3↓<br>1.10↓<br>2.20↓<br>3.30 | 3.1.10.2.20.3.30 | 3↓<br>1.10.2.20.3.30 | 3..1.10↓<br>↓<br>.2.20...3.30 |

## 2.2 C++ cin and cout

For C++, the input and output functions are cin and cout. Cin will read the appropriate data based on the variables you have given by using the operands to point to the variable >>.

## 2.3 Reading n data

The most common type of test data is this type. The way to deal with this is to read in this n and run n times.

C language

```
int main() {
    int n;
    scanf("%d",&n);
    while (n--) {
        /* read each data */
    }
```

```
    return 0;
}
```

C++

```
int main() {
    int n;
    cin>>n;
    while (n--) {
        /* read each data */
    }
    return 0;
}
```

**Example：**

UVA10406: Vito's family

Boss Vito moved in and lived in the street. He hoped that the distance with each relative would be the shortest.

C Language

```
#include <stdio.h>
#define MAX_R 100

int num[MAX_R];

int main() {
    int n,r,s,i;
    int sum;

    scanf("%d",&n);
    while (n--) {
        scanf("%d",&r);
        for (i=0;i<r;i++) {
            scanf("%d",&s);
            num[i]=s;
        }

        //The rest is left for you to complete, hint: median
```

```
        printf("%d\n",sum);
    }

    return 0;
}
```

C++

```
#include <iostream>
#include <vector>
using namespace std;

vector<int> num;

int main() {
    int n,r,s;

    cin>>n;
    while (n--) {
        cin>>r;
        num.clear();
        for (int i=0;i<r;i++) {
            cin>>s;
            num.push_back(s);
        }

        //The rest is left for you to complete, hint: median

        cout<<endl;
    }

    return 0;
}
```

In fact, there may be many layers of "reading n data" for different questions.

**Practice：**

UVA10401: Fibonaccimal Base

UVA10403: Funny Encryption Method

UVA10408: What is the Probability?

## 2.4 Reading to the end of the file

This kind of test data will not tell you how many inputs are there. You must keep dealing until there is no data.

C language

Check the scanf function's return value to see if the data ends. Scanf will return the number of elements it successfully read; when the end of the file is read, scanf returns EOF. To integrate it in the while condition, it becomes the following program:

```
int main() {
    int x;
    while (scanf("%d",&x)!=EOF) {
        /* Process current data */
    }
    return 0;
}
```

C++

If you put cin into the while condition, cin will automatically convert to void*. When the file ends, its value becomes NULL which represents 0 and false. So we write the following program:

```
int main() {
    int x;
    while (cin>>x) {
        /* Process current data */
    }
    return 0;
}
```

**Example：**

UVA10407: Hashmat the brave warrior

Calculate the difference between the number of enemy troops and ours.

C Language

```
#include <stdio.h>
```

```
int main() {
    long long a,b; // May be equal to 2^32, so unsigned
                   // int is not enough
    while (scanf("%lld%lld",&a,&b)!=EOF) {
        /* Try to complete it! */
    }
    return 0;
}
```

Please note that scanf uses %lld for long long.

C++

```
#include <iostream>
using namespace std;

int main() {
    long long a,b; // May be equal to 2^32, so unsigned
                   // int is not enough
    while (cin>>a>>b) {
      /* Try to complete it! */
    }
    return 0;
}
```

**Prcatice：**

10400: The 3n + 1 problem

10405: Jolly Jumpers (Mixed two ways)

10411: Back to High School Physics

## 2.5 End until read 0

This type is also a frequently occurring pattern. Simply add a condition for jumping out of the loop.

C language

```c
int main() {
    int n;
    while (scanf("%d",&n)!=EOF) {
        if (n==0) break;

        /* ... */
    }
    return 0;
}
```

C++

```cpp
int main() {
    int n;
    while (cin>>n) {
        if (n==0) break;

        /* ... */
    }
    return 0;
}
```

**Examle：**

UVA10404: Primary Arithmetic

Calculate the number of times to carry out the addition.

C Language

```c
#include <stdio.h>

int main() {
    int a,b;
    while (scanf("%d%d",&a,&b)!=EOF) {
        if (a==0&&b==0) break;

        /* ... */
    }
    return 0;
}
```

C++

```
#include <iostream>
using namespace std;

int main() {
    int a,b;
    while (cin>>a>>b) {
        if (a==0&&b==0) break;
        /* ... */
    }
    return 0;
}
```

**Practice：**

10416: Last Digit

10418: Minesweeper

## 2.6 Read a list of data at a time until the end of the file

A row of data may contains blanks and cannot be handled as normal strings, so an entire row of data needs to be read at a time. You can use fgets in C, or getline in C++.

C language

```
#include <stdio.h>
int main() {
    char s[100];
    while ( fgets(s,100,stdin) != NULL ) { // 100 is column's size
        /* ... */
    }
    return 0;
}
```

C++

```
#include <cstring>
int main() {
    string s;
    while ( getline(cin,s) ) {
```

```
        /* ... */
    }
    return 0;
}
```

**Example：**

UVA272: [TEX Quotes]

Replace paired double quotes ” and ” with `` and '' 。

C language

```c
int main() {
    char s[100005];
    while ( fgets(s,100005,stdin) != NULL ) {
        /* ... */
    }
    return 0;
}
```

C++

```cpp
int main() {
    string s;
    while ( getline(cin,s) ) {
        /* ... */
    }
    return 0;
}
```