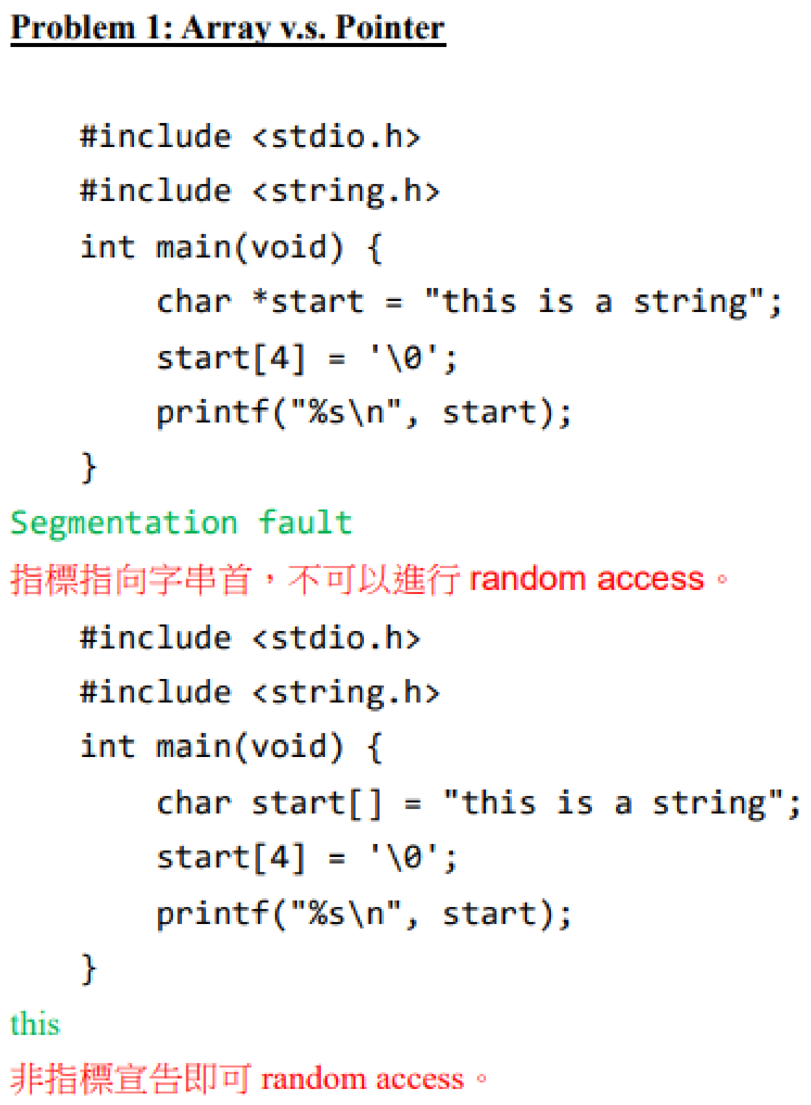
C / C++ Common Bugs

Presenter: Yu-Cheng Chen

Date: 2023/8/9

Problem 1



Problem 1: Array v.s. Pointer

```
#include <stdio.h>
#include <string.h>
int main(void) {
    char *start = "this is a string";
    start[4] = '\0';
    printf("%s\n", start);
}
```

```
Segmentation fault
```

指標指向字串首，不可以進行 random access。

```
#include <stdio.h>
#include <string.h>
int main(void) {
    char start[] = "this is a string";
    start[4] = '\0';
    printf("%s\n", start);
}
```

this

非指標宣告即可 random access。

Problem 2

```
#include <stdio.h>
#include <string.h>
int main() {
  char string[] = "this is a string";
  char *start;
  start = string;
  start = strtok(start, " ");
  while (start != NULL) {
    printf("%s\n", start);
    start = strtok(NULL, " ");
  }
  start = string;
  start = strtok(start, " ");
  while (start != NULL) {
    printf("%s\n", start);
    start = strtok(NULL, " ");
  }
}
```

```
this
is
a
string
```

start 指標已經在第一個 loop 被 strtok 切開，
第二個 loop 讀就只剩下切開後的第一 part。

```
this
```

Problem 3

Explain why a character is missing.

```
#include <stdio.h>
int main() {
  FILE *fp = fopen("file", "wb");
  for (int i = 0; i < 256; i++)
    fputc(i, fp);
  fclose(fp);
  fp = fopen("file", "rb");
  int count = 0;
  char c;
  while ((c = fgetc(fp)) != EOF)
    count++;
  printf("count = %d\n", count);
}
```

因為數字 255 寫進檔案裡是 FF，代表 EOF。所以不會被底下 while 迴圈吃進去。

Problem 4

Why is our lab number incorrect?

```
#include <stdio.h>
int main() {
  long int lab_tel = 035731603;
  printf("my lab's telephone number is %ld\n", lab_tel);
}
```

my lab's telephone number is 7844739

前面多加了一個 0。C++中以 0 開頭表示八進制。35731603 轉成十進制即 7844739。

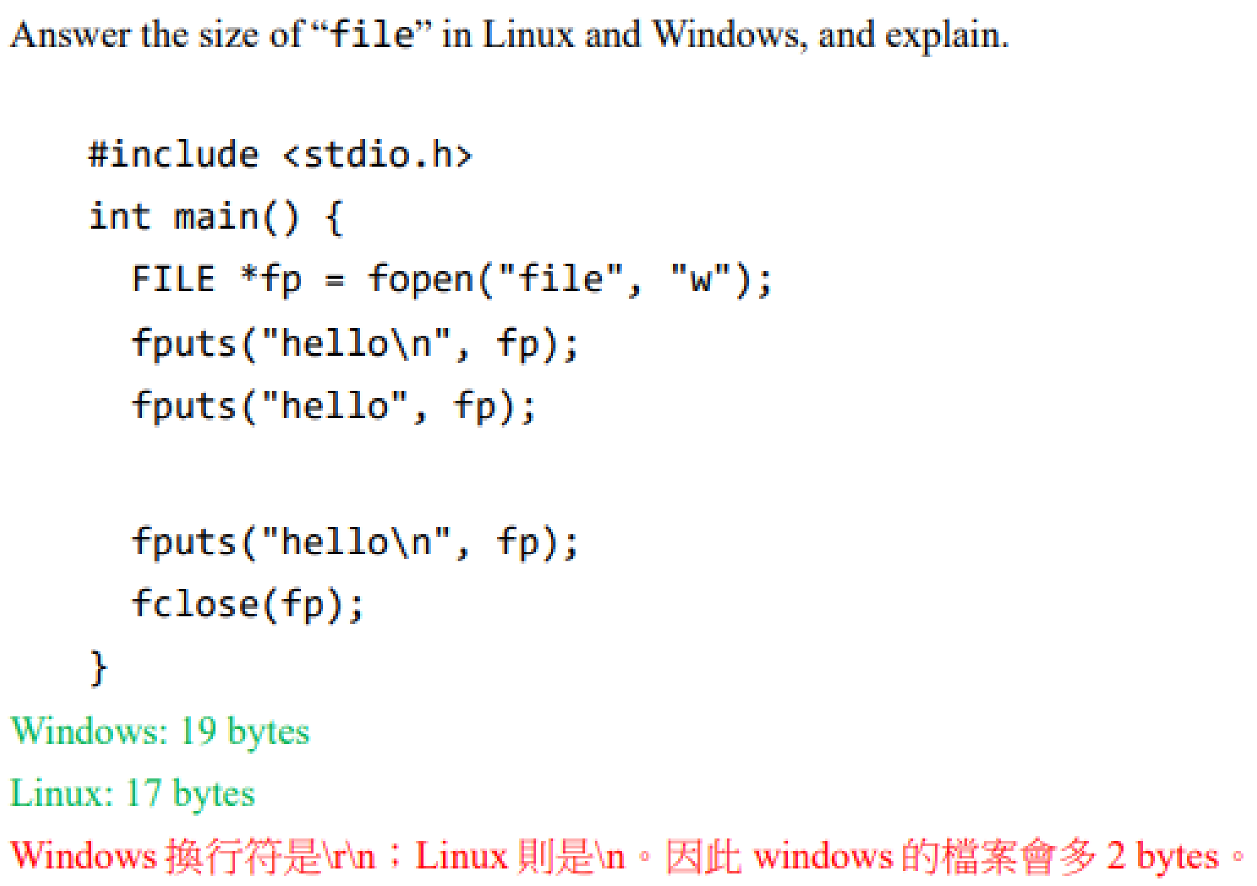
Problem 5

```
#include <stdio.h>
int main() {
  int a[10];
  if (a == &a)
    printf("yes\n");
  else
    printf("no\n");

  if (a + 1 == &a + 1)
    printf("yes\n");
  else
    printf("no\n");
}
```

error: comparison between distinct pointer types 'int*' and 'int (*)[10]' lacks a cast [-fpermissive]

在紅字行&前或後加*即可跑出 yes。

Problem 6



Answer the size of “`file`” in Linux and Windows, and explain.

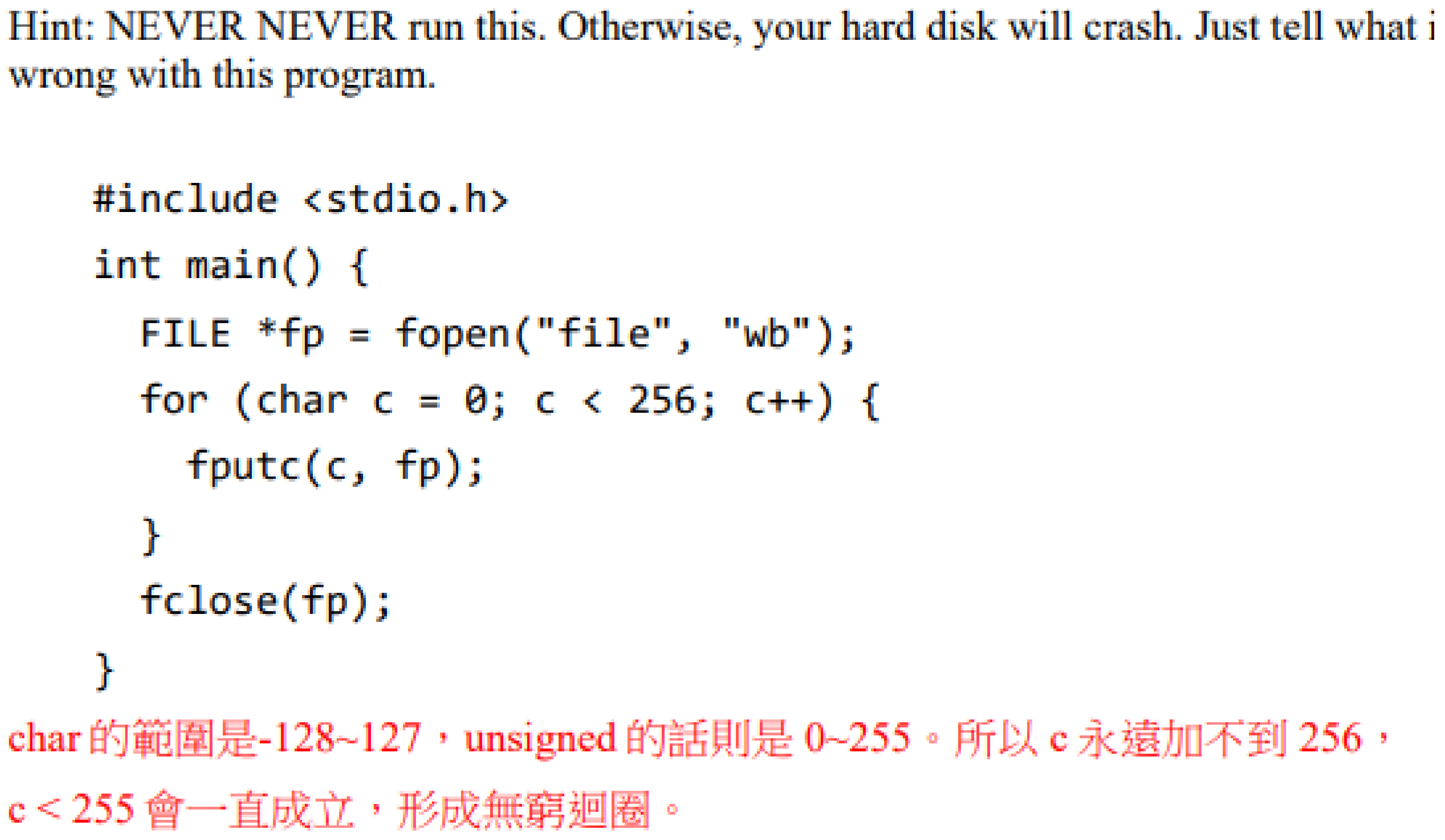
```
#include <stdio.h>
int main() {
  FILE *fp = fopen("file", "w");
  fputs("hello\n", fp);
  fputs("hello", fp);

  fputs("hello\n", fp);
  fclose(fp);
}
```

Windows: 19 bytes

Linux: 17 bytes

Windows 換行符是\r\n；Linux 則是\n。因此 windows 的檔案會多 2 bytes。

Problem 7



Hint: NEVER NEVER run this. Otherwise, your hard disk will crash. Just tell what i
wrong with this program.

```
#include <stdio.h>
int main() {
  FILE *fp = fopen("file", "wb");
  for (char c = 0; c < 256; c++) {
    fputc(c, fp);
  }
  fclose(fp);
}
```

char 的範圍是-128~127，unsigned 的話則是 0~255。所以 c 永遠加不到 256，
c < 255 會一直成立，形成無窮迴圈。

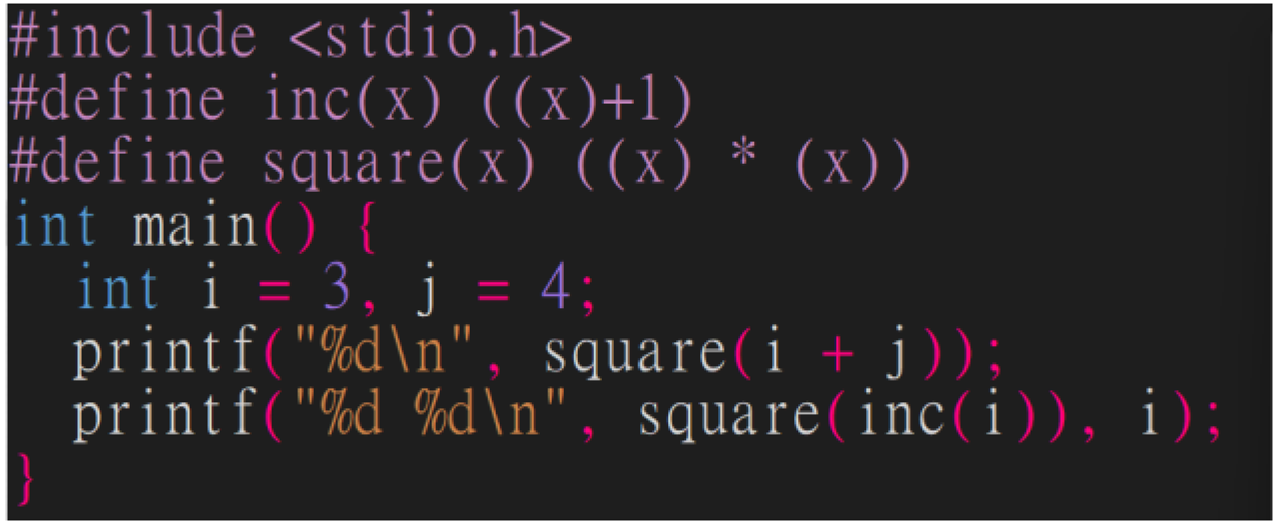
Problem 8

```
#include <stdio.h>
#define inc(x) ((x)++)
#define square(x) (x * x)
int main() {
  int i = 3, j = 4;
  printf("%d\n", square(i + j));
  printf("%d %d\n", square(inc(i)), i);
}
```

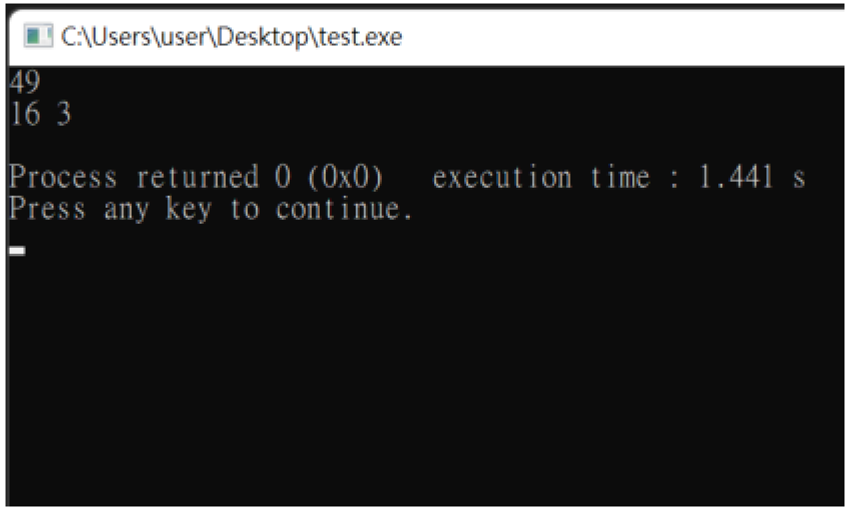
19

12 3

巨集要在變數前後加括號，且不能用遞增、遞減運算子。上述 code 應改成



```
#include <stdio.h>
#define inc(x) ((x)+1)
#define square(x) ((x) * (x))
int main() {
  int i = 3, j = 4;
  printf("%d\n", square(i + j));
  printf("%d %d\n", square(inc(i)), i);
}
```



C:\Users\user\Desktop\test.exe

```
49
16 3

Process returned 0 (0x0)   execution time : 1.441 s
Press any key to continue.
```

Problem 9

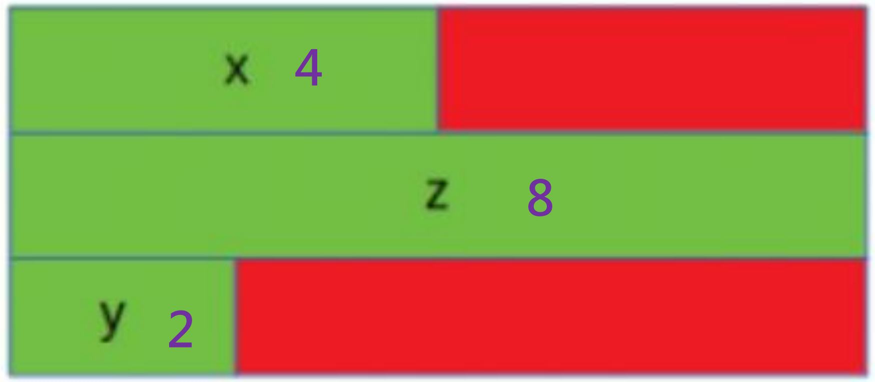
Is sizeof for a struct equal to the sum of sizeof of each member? - GeeksforGeeks

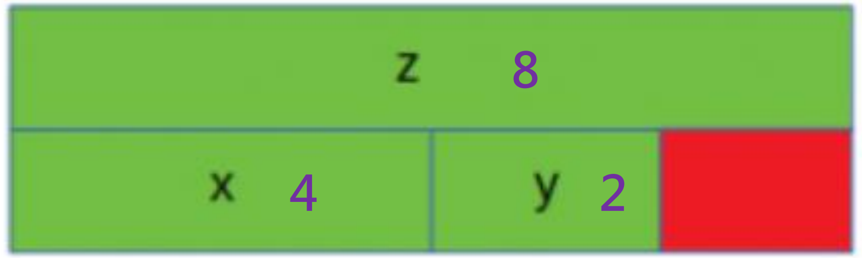
```c
#include <stdio.h>
struct csie {
  char c;     // 1
  short s;    // 2
  int i;      // 4
  double e;   // 8
};
struct ceis {
  char c;     // 1
  double e;   // 8
  int i;      // 4
  short s;    // 2
};
int main() {
  printf("csie = %d\n", sizeof(struct csie));
  printf("ceis = %d\n", sizeof(struct ceis));
}
```

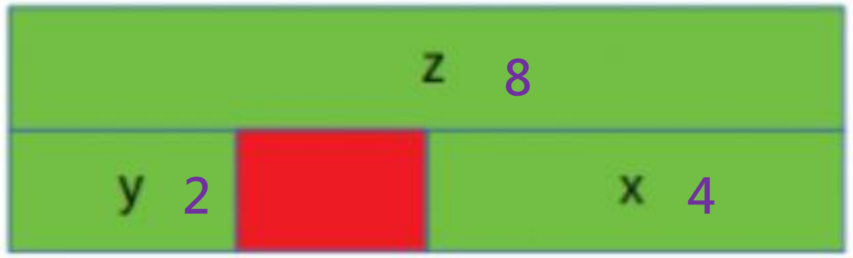
csie = 16

ceis = 24

compiler 會自動對齊(padding)。







Problem 10

```c
#include <stdio.h>
#include <string.h>
int main() {
  char source[] = "This is a string.";

  char destination[4];
  int i = 5;
  strcpy(destination, source);
  printf("i is %d\n", i);
  printf("source is [%s]\n", source);
  printf("destination is [%s]\n", destination);
}
```

```
i is 5
source is [ is a string.]
destination is [This is a string.]
```

dest 不夠長會有 buffer overflow 問題。

```
On stack memory:

low address                                 high address
    |                                           |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+--+
    | | | | |T|h|i|s| |i|s| |a| |s|t|r|i|n|g|.|\0|
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+--+
             ^
           source

after strcpy()

    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+--+-+-+-+--+
    |T|h|i|s| |i|s| |a| |s|t|r|i|n|g|.|\0|n|g|.|\0|
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+--+-+-+-+--+
     ^       ^
  target   source
```

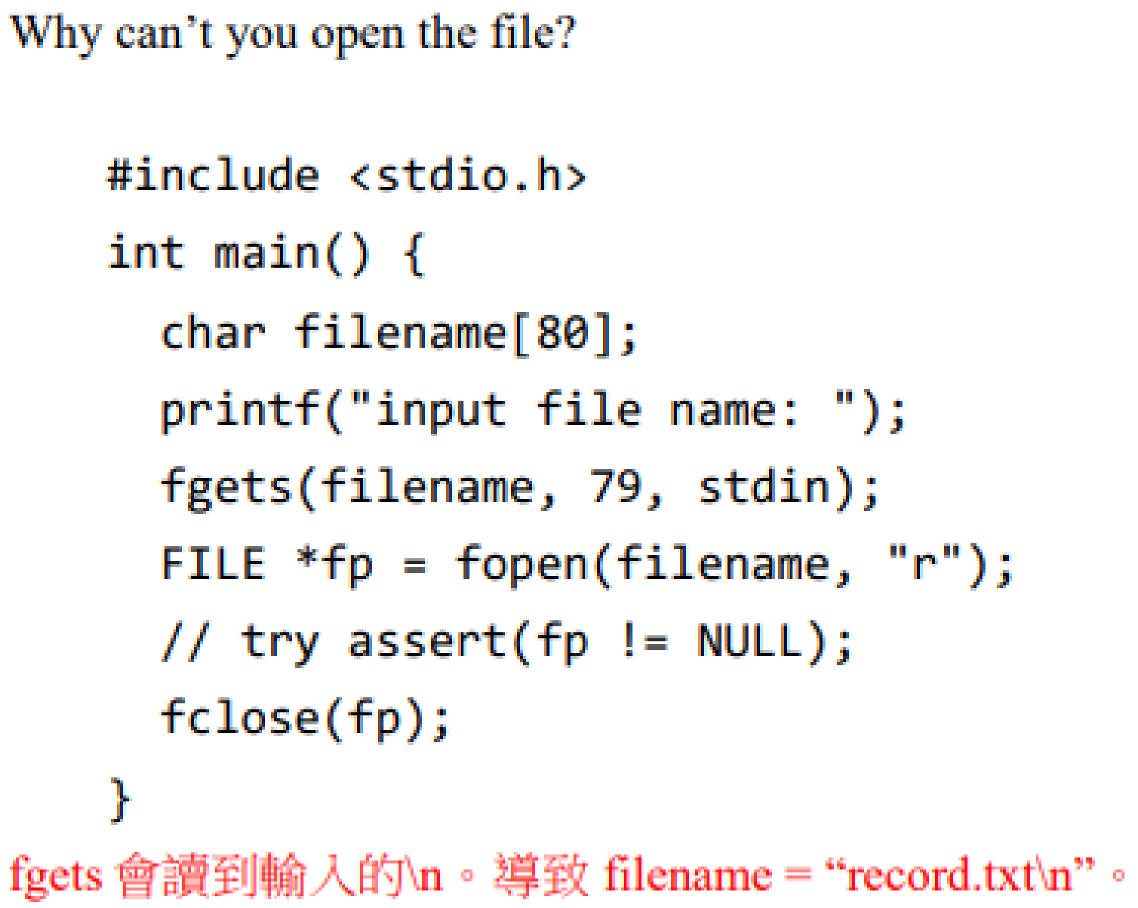
Problem 11

```
// header.h
#include <stdio.h>
static int val = 0;
void set(int x);

// impl.c
#include "header.h"
void set(int x) {
  val = x;
}

// main.c
#include "header.h"
int main() {
  set(100);
  if (val == 100)
    printf("val == 100\n");
  else
    printf("val != 100\n");
}
```

val != 100

static int 在一個 c 檔會產生一個值，兩個 c 檔就會產生兩個值，而這兩個值不會同步。要同步則需宣告 extern。

Problem 12

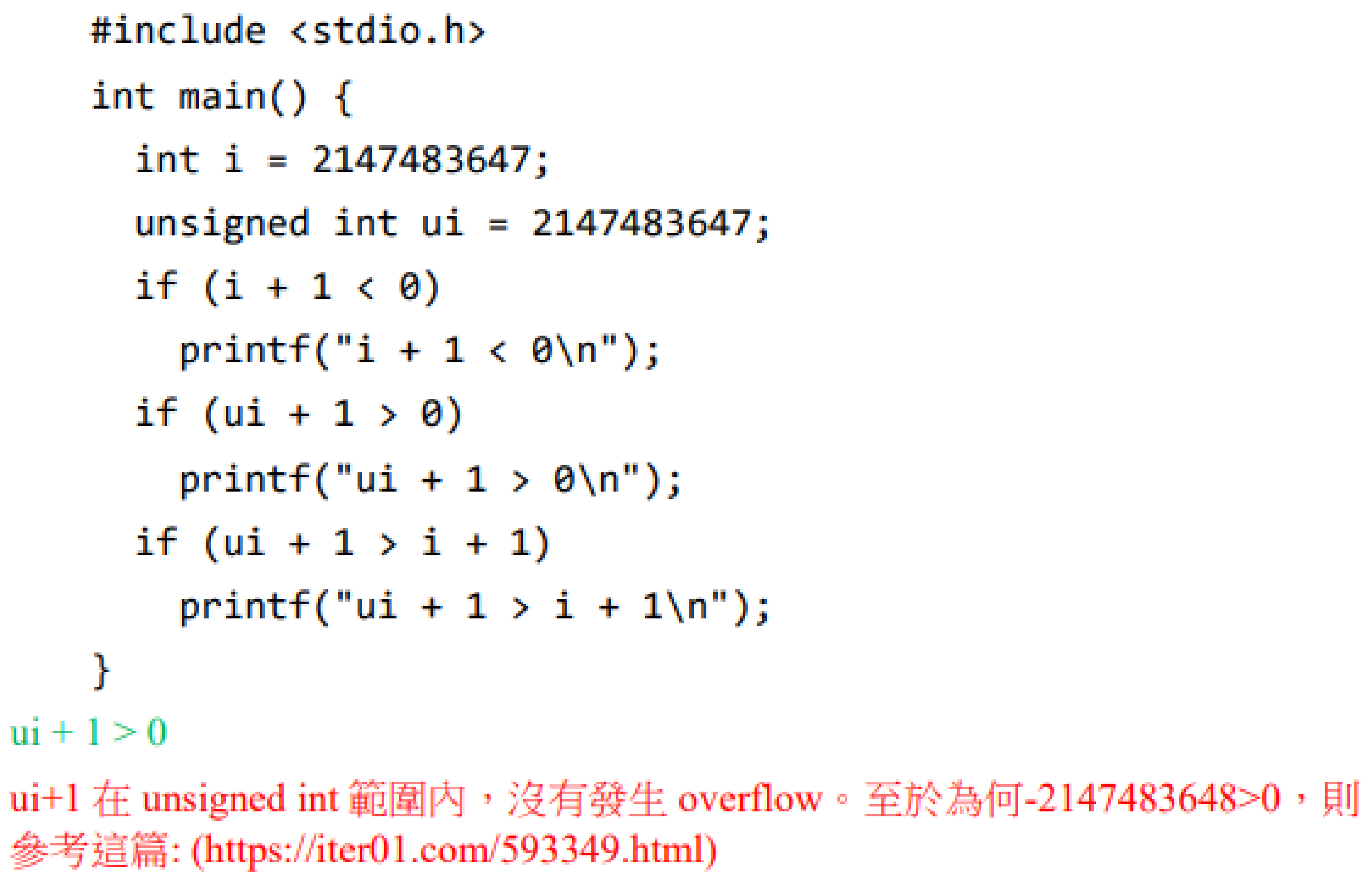


Why can’t you open the file?

```
#include <stdio.h>
int main() {
  char filename[80];
  printf("input file name: ");
  fgets(filename, 79, stdin);
  FILE *fp = fopen(filename, "r");
  // try assert(fp != NULL);
  fclose(fp);
}
```

fgets 會讀到輸入的\n。導致 filename = “record.txt\n”。

Problem 13



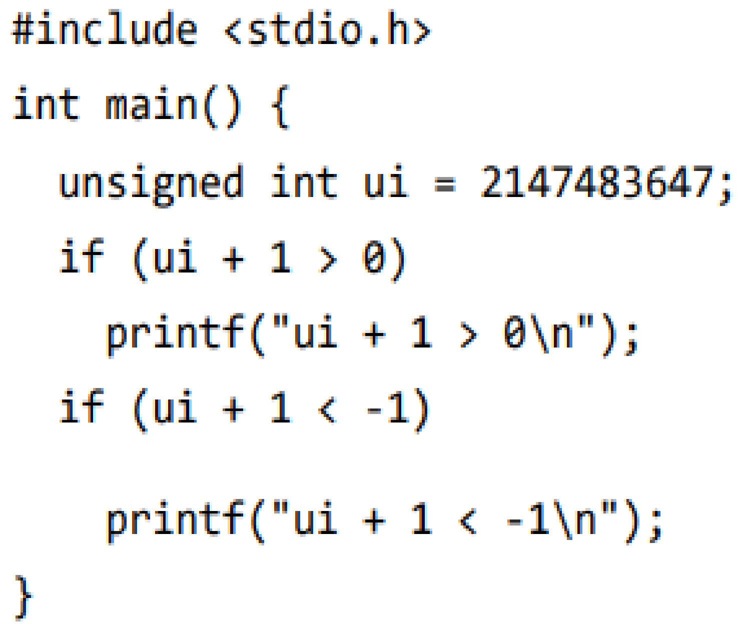
```
#include <stdio.h>
int main() {
  int i = 2147483647;
  unsigned int ui = 2147483647;
  if (i + 1 < 0)
    printf("i + 1 < 0\n");
  if (ui + 1 > 0)
    printf("ui + 1 > 0\n");
  if (ui + 1 > i + 1)
    printf("ui + 1 > i + 1\n");
}
```

ui + 1 > 0

ui+1 在 unsigned int 範圍內，沒有發生 overflow。至於為何-2147483648>0，則參考這篇: (https://iter01.com/593349.html)

C/C++ 中的算術及其陷阱 | IT人 (iter01.com)

Problem 14



```
#include <stdio.h>
int main() {
  unsigned int ui = 2147483647;
  if (ui + 1 > 0)
    printf("ui + 1 > 0\n");
  if (ui + 1 < -1)

    printf("ui + 1 < -1\n");
}
```

```
ui + 1 > 0
ui + 1 < -1
```

當一個 signed int 和一個 unsigned int 進行比較時，signed int 會被自動轉型成 unsigned int 進行比較。因此 -1 => 4294967295

Problem 15

```c
#include <stdio.h>
int main() {
  int i = -13;
  if ((i / 2) == (i >> 1))
    printf("yes\n");
  else
    printf("no\n");
}
```

no

```
i / 2 == -6
i >> 1 == -7
```

Problem 16

```c
#include <stdio.h>
#include <stdlib.h>
int compare(const void *a, const void *b) {
  return (*(int *)a - *(int *)b);
}

int main() {
  int values[] = {-2147483640, 50, 100};
  qsort(values, 3, sizeof(int), compare);
  for (int n = 0; n < 3 ; n++)
     printf("%d ", values[n]);
}
```

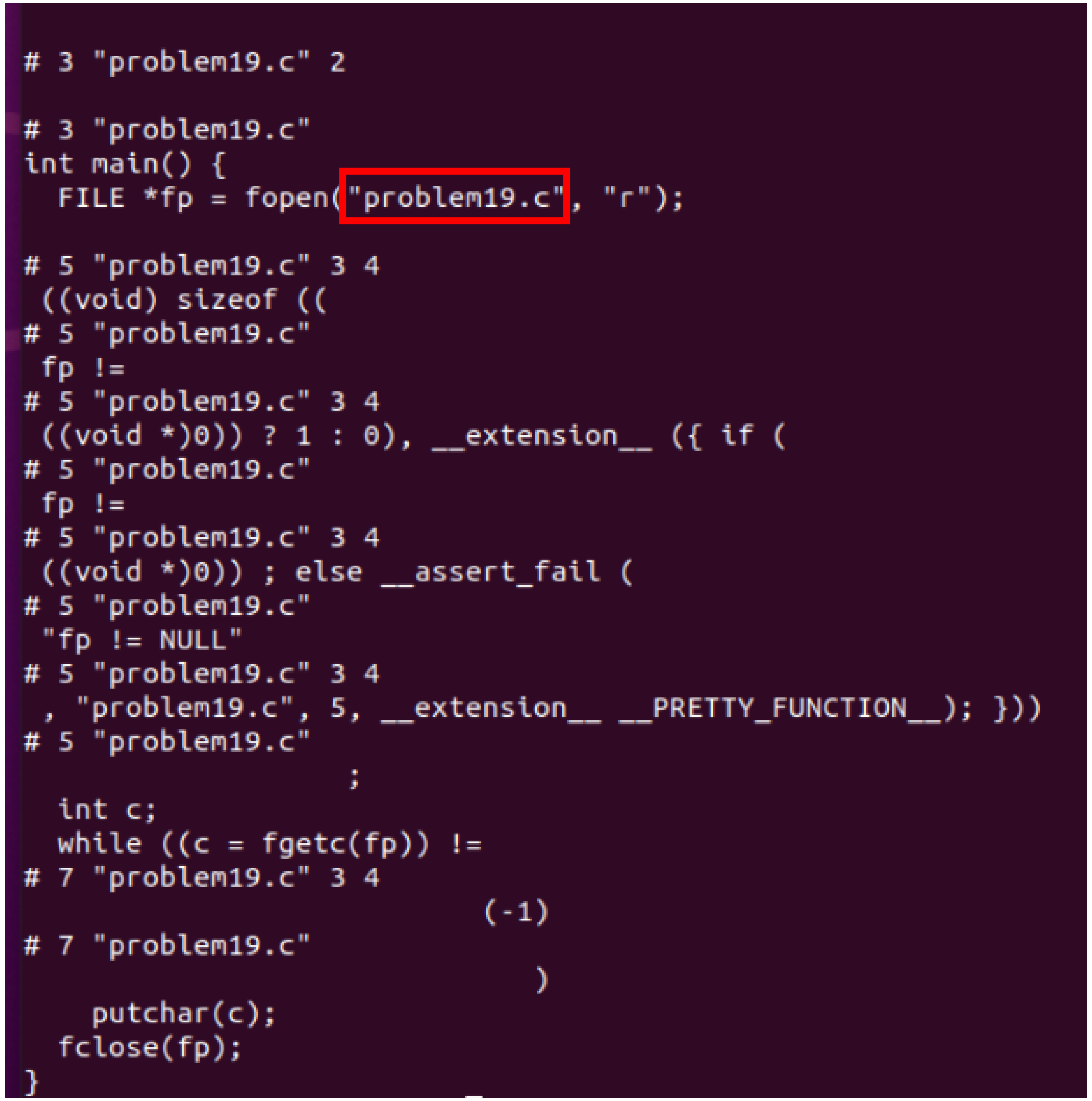
```
50 100 -2147483640
```

因為 a-b 會發生 overflow。

Problem 17

```
#include <stdio.h>
#include <assert.h>
int main() {
  FILE *fp = fopen(__FILE__, "r");
  assert(fp != NULL);
  int c;
  while ((c = fgetc(fp)) != EOF)
    putchar(c);
  fclose(fp);
}
```

Output 會印出自己這份檔案的內容。因為__FILE__是 compiler 預先定義好的變數，表示自己這份檔案。因此上述 code 做的事就是自己讀自己然後印出來。使用-E (下圖)就能在第 4 行看到 fopen 直接被編譯器轉成檔名:



```
# 3 "problem19.c" 2

# 3 "problem19.c"
int main() {
  FILE *fp = fopen("problem19.c", "r");

# 5 "problem19.c" 3 4
 ((void) sizeof ((
# 5 "problem19.c"
 fp !=
# 5 "problem19.c" 3 4
 ((void *)0)) ? 1 : 0), __extension__ ({ if (
# 5 "problem19.c"
 fp !=
# 5 "problem19.c" 3 4
 ((void *)0)) ; else __assert_fail (
# 5 "problem19.c"
 "fp != NULL"
# 5 "problem19.c" 3 4
 , "problem19.c", 5, __extension__ __PRETTY_FUNCTION__); }))
# 5 "problem19.c"
                   ;
  int c;
  while ((c = fgetc(fp)) !=
# 7 "problem19.c" 3 4
                           (-1)
# 7 "problem19.c"
                              )
    putchar(c);
  fclose(fp);
}
```

Problem 18

```c
#include <stdio.h>
#define SWAP(x, y) x ^= y ^= x ^= y
int main() {
  int i = 3, j = 5;
  printf("%d %d\n", i, j);
  SWAP(i, j);
  printf("%d %d\n", i, j);
  SWAP(i, i);
  printf("%d\n", i);
}
```

```
3 5
5 3
0
```

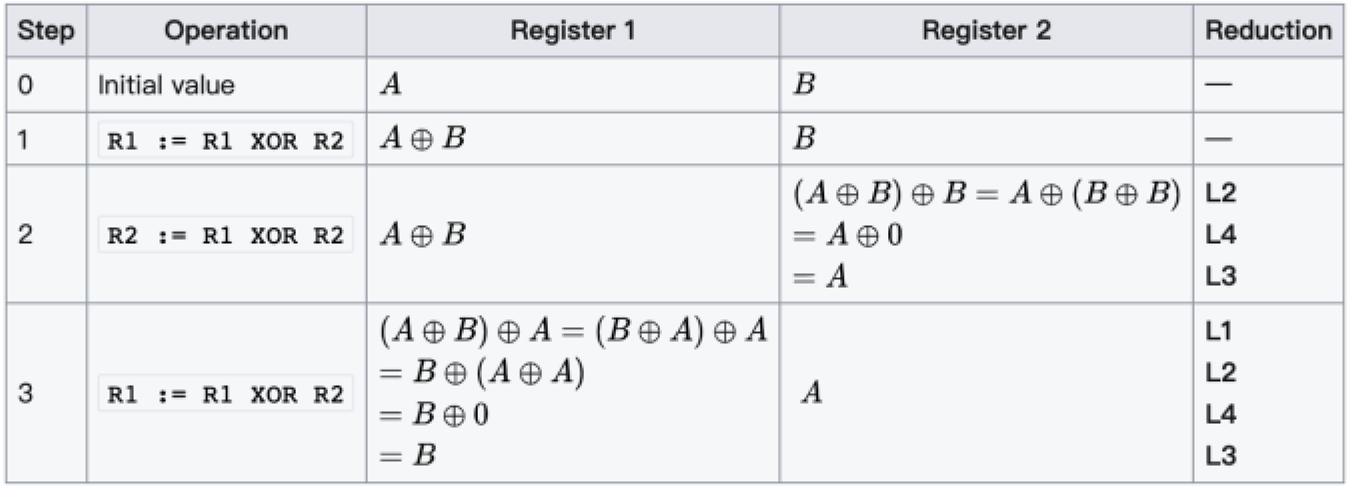
先做 x^=y。此時 x=6, y=5。 (x = x XOR y)

再做 y^=x。此時 x=6, y=3。

最後 x^=y。此時 x=5, y=3。

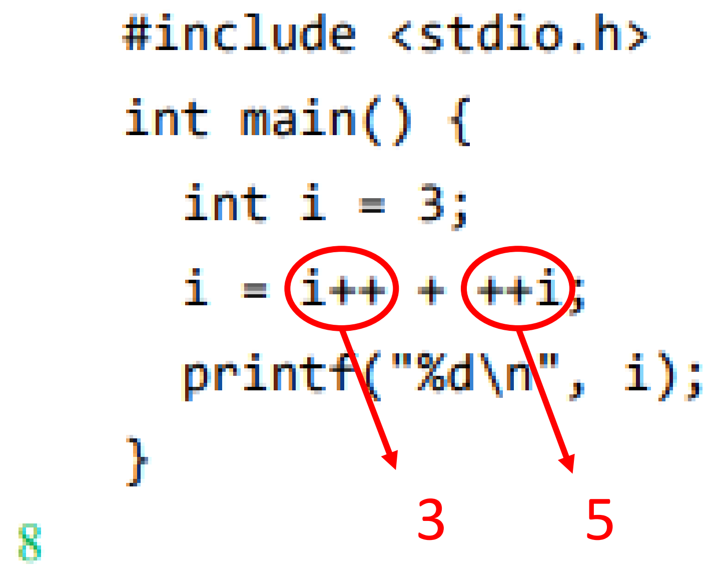
自己和自己做 XOR 則會變 0。

XOR Swap Algorithm:



Step	Operation	Register 1	Register 2	Reduction
0	Initial value	A	B	—
1	`R1 := R1 XOR R2`	$A \oplus B$	B	—
2	`R2 := R1 XOR R2`	$A \oplus B$	$(A \oplus B) \oplus B = A \oplus (B \oplus B)$ $= A \oplus 0$ $= A$	L2 L4 L3
3	`R1 := R1 XOR R2`	$(A \oplus B) \oplus A = (B \oplus A) \oplus A$ $= B \oplus (A \oplus A)$ $= B \oplus 0$ $= B$	A	L1 L2 L4 L3

Problem 19



```
#include <stdio.h>
int main() {
  int i = 3;
  i = i++ + ++i;
  printf("%d\n", i);
}
```

```
8
```

Problem 20

```c
#include <stdio.h>
int *bar(int t) {
  int i = t;
  int *temp = &i;
  printf("temp is %d, (*temp) is %d\n", temp, *temp);
  return temp;
}

void foo(int a, int b) {
  int i;
  int *temp = &i;
  *temp = a + b;
}

int main() {
  int *a;
  a = bar(10);
  printf("a is %d, (*a) is %d \n", a, *a);
  foo(10, 20);
  printf("a is %d, (*a) is %d \n", a, *a);
}
```

```
temp is 6422248, (*temp) is 10

a is 6422248, (*a) is 10
a is 6422248, (*a) is 6422476
```

local 變數的生命週期只會在函式內。因此呼叫了 foo 後就會把原本 a 的位址蓋掉了。

Problem 21

```
#include <stdio.h>
int main() {
  char i = 1;
  char j;
  scanf("%d", &j);
  if (i & j)
    printf("yes.\n");
  else
    printf("no.\n");
}
```

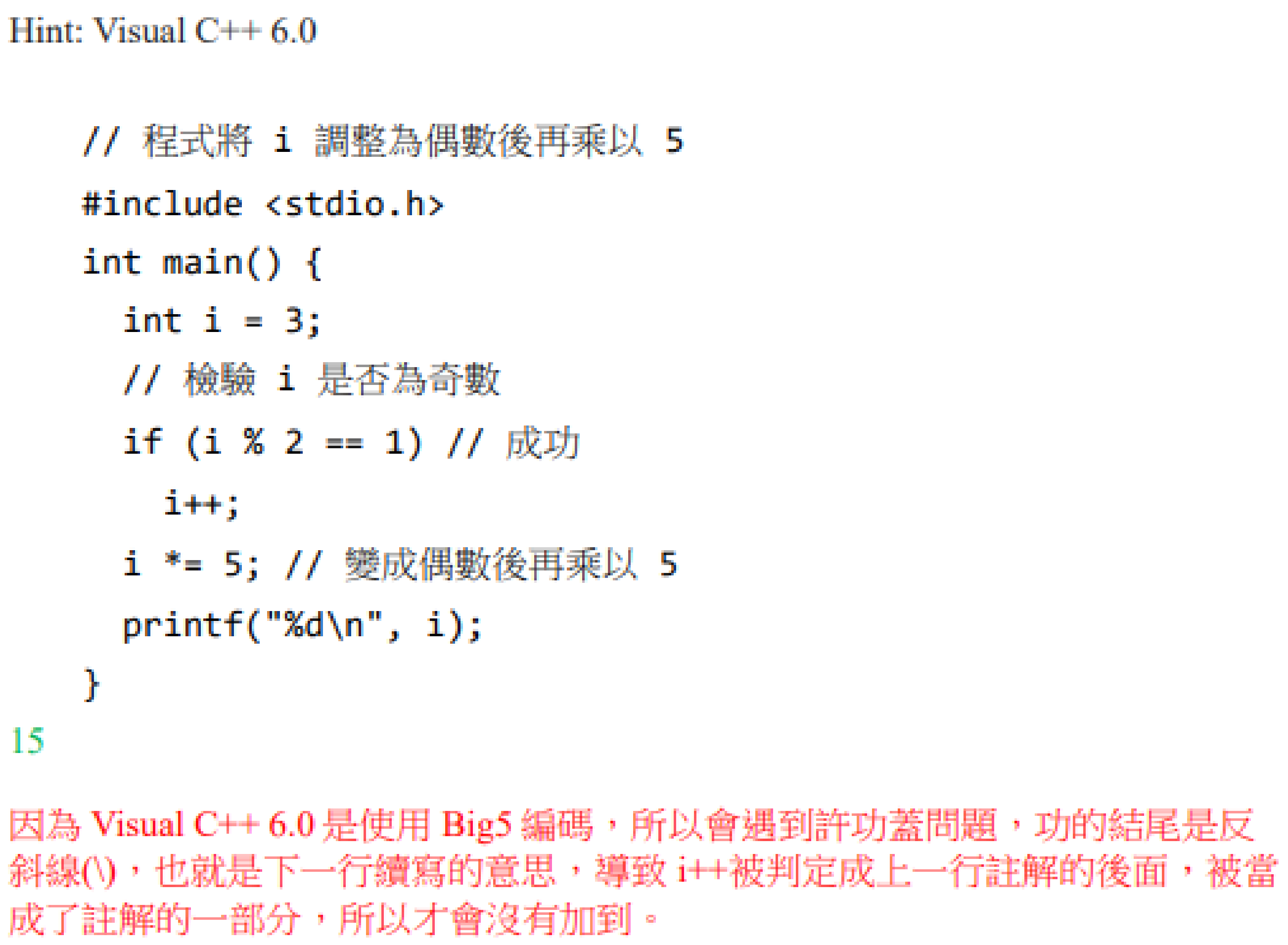
Input:

3

no.

因為 char 分配給 j 的空間只有 1byte，而 scanf(“%d”)卻配了 4byte 的 int 空間給 j。導致其侵占到了 i 的空間。使 i 的空間裡的值變成 0。

Problem 22



Hint: Visual C++ 6.0

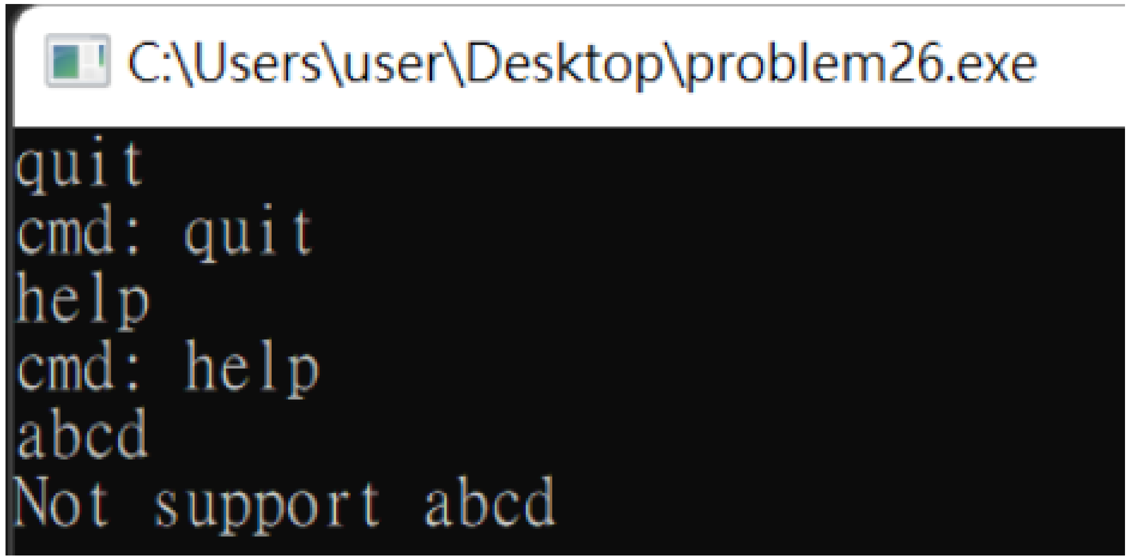
```
// 程式將 i 調整為偶數後再乘以 5
#include <stdio.h>
int main() {
  int i = 3;
  // 檢驗 i 是否為奇數
  if (i % 2 == 1) // 成功
    i++;
  i *= 5; // 變成偶數後再乘以 5
  printf("%d\n", i);
}
```

15

因為 Visual C++ 6.0 是使用 Big5 編碼，所以會遇到許功蓋問題，功的結尾是反斜線(\)，也就是下一行續寫的意思，導致 i++被判定成上一行註解的後面，被當成了註解的一部分，所以才會沒有加到。

Problem 23

```cpp
#include <cstdio>
#include <iostream>
#include <map>
#include <string>
#define FuncDef(cmd) void cmd_##cmd() { printf("cmd: "#cmd"\n"); }
#define RegFunc(cmd) m[#cmd] = cmd_##cmd;

std::map<std::string, void(*)()> m;
FuncDef(quit);
FuncDef(help);
int main() {
  RegFunc(quit);
  RegFunc(help);
  std::string cmd;
  while (getline(std::cin, cmd)) {
    if (m.count(cmd)) (*m[cmd])();
    else printf("Not support %s\n", cmd.c_str());
  }
}
```


C:\Users\user\Desktop\problem26.exe

```
quit
cmd: quit
help
cmd: help
abcd
Not support abcd
```

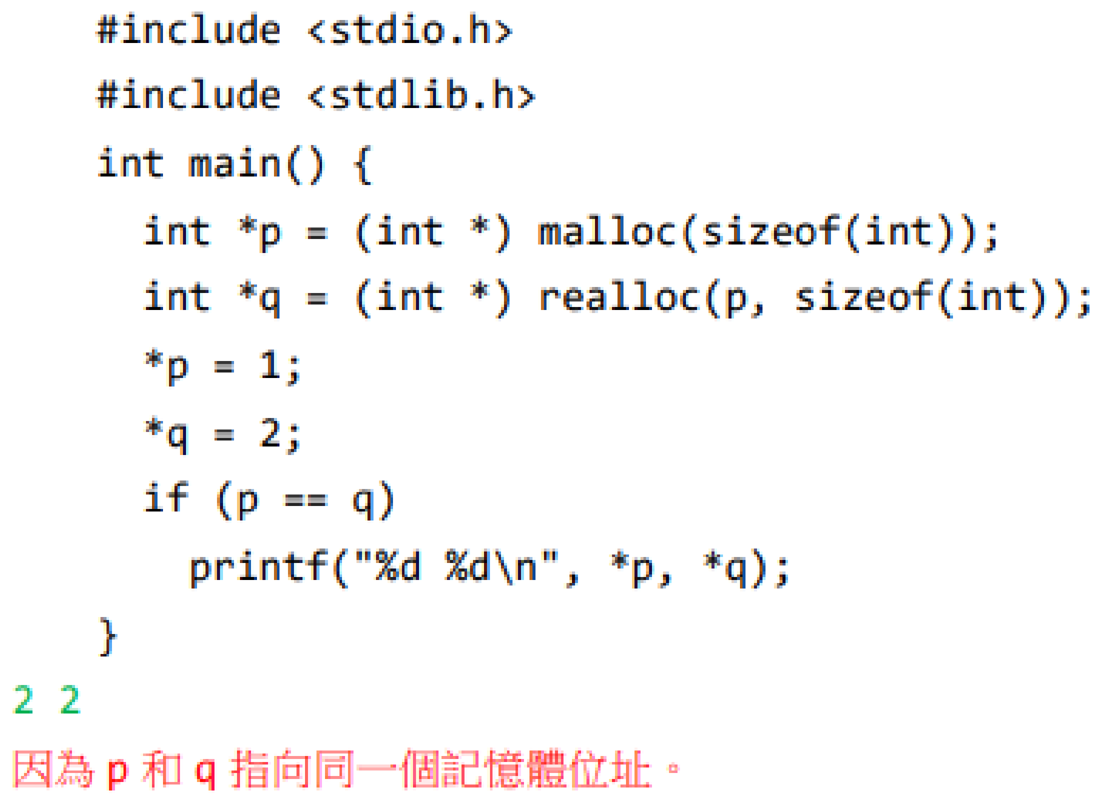
- Macro '#' and "##"
  - '#' means replace target with "string" (with double quote)

    ```c
    #define mkstr(s) #s
    printf(mkstr(geeksforgeeks)); // printf("geeksforgeeks");
    ```

  - "##" means replace target with str (plain text replacement)

    ```c
    #define concat(a, b) a##b
    int xy = 30;
    printf("%d", concat(x, y)); // concat(x, y) = xy
    ```

Problem 24

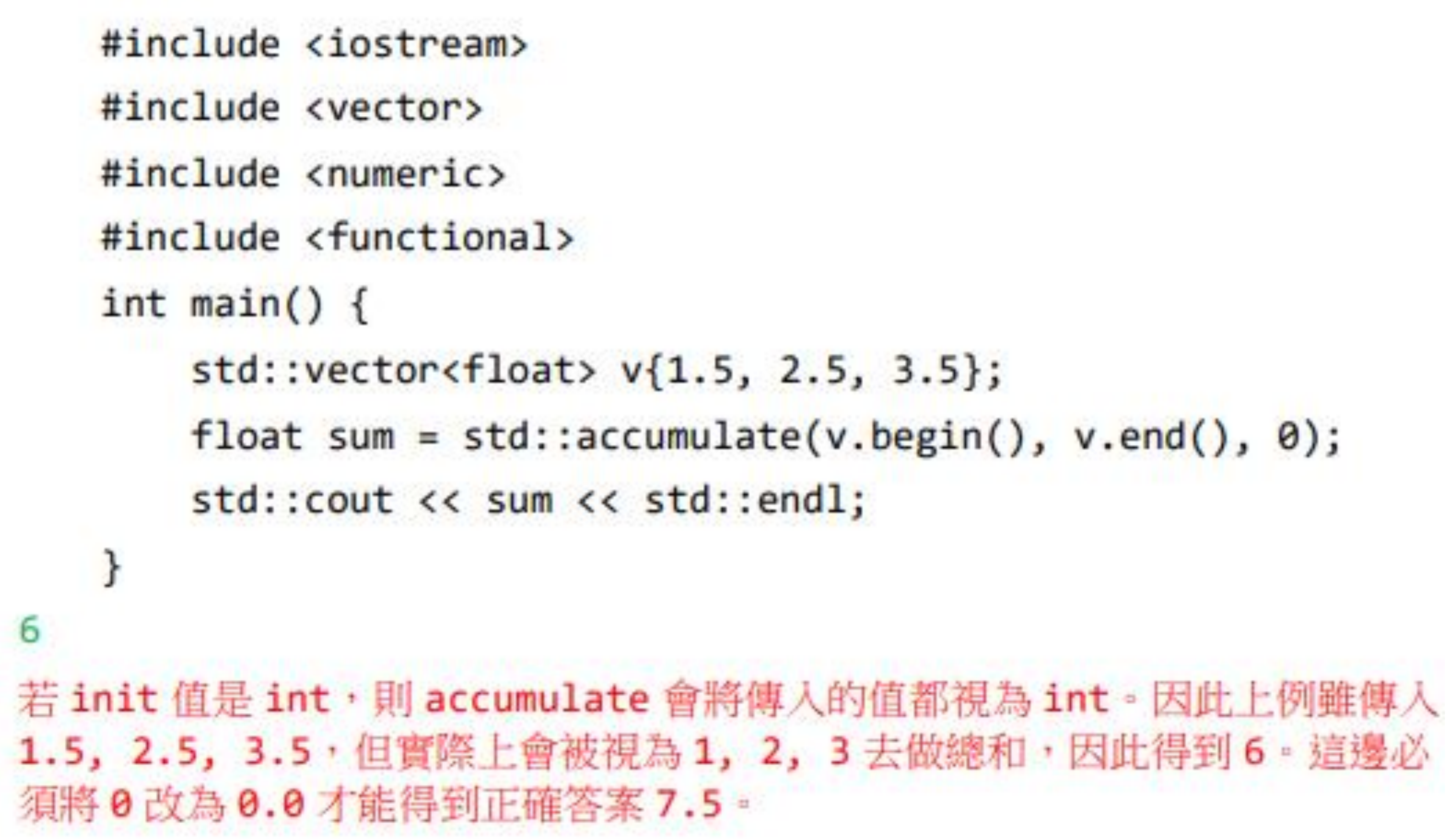


```c
#include <stdio.h>
#include <stdlib.h>
int main() {
  int *p = (int *) malloc(sizeof(int));
  int *q = (int *) realloc(p, sizeof(int));
  *p = 1;
  *q = 2;
  if (p == q)
    printf("%d %d\n", *p, *q);
}
```

```
2 2
```

因為 p 和 q 指向同一個記憶體位址。

Problem 25

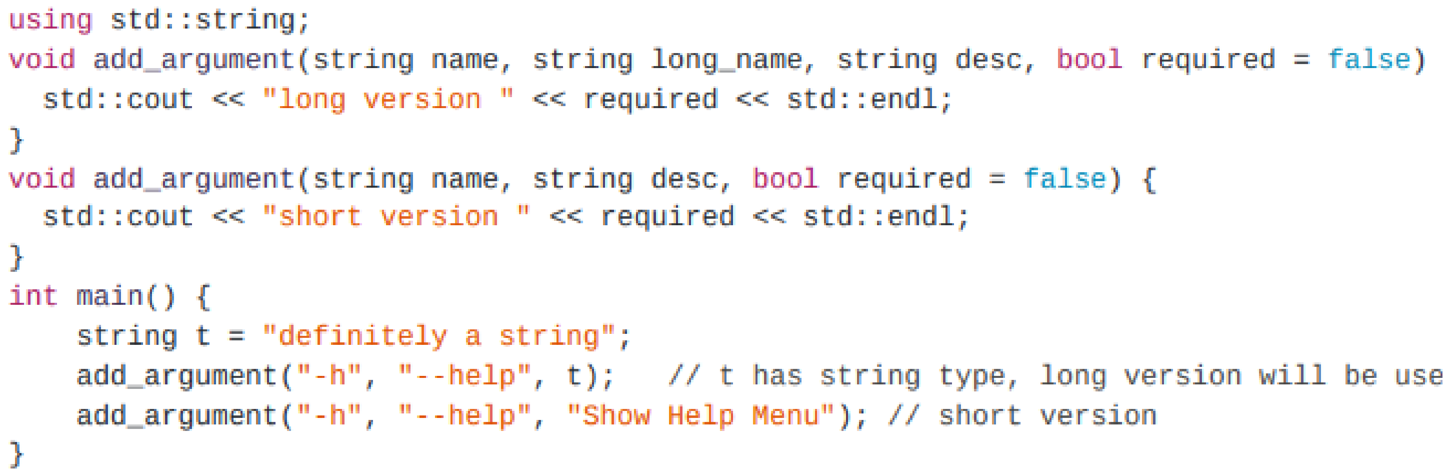


```
#include <iostream>
#include <vector>
#include <numeric>
#include <functional>
int main() {
    std::vector<float> v{1.5, 2.5, 3.5};
    float sum = std::accumulate(v.begin(), v.end(), 0);
    std::cout << sum << std::endl;
}
```

```
6
```

若 init 值是 int，則 accumulate 會將傳入的值都視為 int。因此上例雖傳入 1.5, 2.5, 3.5，但實際上會被視為 1, 2, 3 去做總和，因此得到 6。這邊必須將 0 改為 0.0 才能得到正確答案 7.5。

Problem 26



```
using std::string;
void add_argument(string name, string long_name, string desc, bool required = false)
  std::cout << "long version " << required << std::endl;
}
void add_argument(string name, string desc, bool required = false) {
  std::cout << "short version " << required << std::endl;
}
int main() {
    string t = "definitely a string";
    add_argument("-h", "--help", t);   // t has string type, long version will be use
    add_argument("-h", "--help", "Show Help Menu"); // short version
}
```

```
    #include <iostream>
    #include <string>
    using std::string;
    void add_argument(string name, string long_name, string
desc, bool required = false) {
      std::cout << "long version " << required << std::endl;
    }
    void add_argument(string name, string desc, bool required =
false) {
      std::cout << "short version " << required << std::endl;
    }
    int main() {
      add_argument("-h", "--help", "Show Help Menu");
    }
```

String constant does not have type string thus it's taken as bool, ended up in the short version function.

short version 1