1. 右側程式正確的輸出應該如下：
2. int k = 4;
3. int m = 1;

3 for (int i=1; i<=5; i=i+1) {

4 for (int j=1; j<=k; j=j+1) {

5 printf (" "); 6 }

7 for (int j=1; j<=m; j=j+1) {

8 printf ("\*"); 9 }

10 printf ("\n"); 11 k = k – 1;

12 m = m + 1;

13 }

\*

\*\*\*

\*\*\*\*\*

\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*

在不修改右側程式之第 4 行及第 7 行程式碼的前提下，最少需修改幾行程式碼以得到正確輸出？

* 1. 1
  2. 2
  3. 3
  4. 4

2. 給定一陣列 **a[10]={ 1, 3, 9, 2, 5,**

int f (int a[], int n) { int index = 0;

for (int i=1; i<=n-1; i=i+1) { if (a[i] >= a[index]) {

index = i;

}

}

return index;

}

**8, 4, 9, 6, 7 }**，i.e., a[0]=1,a[1]=3, …,

a[8]=6, a[9]=7，以 **f(a, 10)**呼叫執行右側函式後，回傳值為何？

1. 1
2. 2
3. 7
4. 9
5. 給定一整數陣列 a[0]、a[1]、…、a[99]且 a[k]=3k+1，以 value=100 呼叫以下兩函式，假設函式 **f1** 及 **f2** 之 **while** 迴圈主體分別執行 n1 與 n2 次 (i.e, 計算 **if** 敘述執行次數，不包含 **else if** 敘述)，請問 n1 與 n2 之值為何? 註： (low + high)/2 只取整數部分。

int f1(int a[], int value) { int r\_value = -1;

int i = 0;

while (i < 100) {

if (a[i] == value) { r\_value = i; break;

}

i = i + 1;

}

return r\_value;

}

int f2(int a[], int value) { int r\_value = -1;

int low = 0, high = 99; int mid;

while (low <= high) { mid = (low + high)/2; if (a[mid] == value) {

r\_value = mid; break;

}

else if (a[mid] < value) { low = mid + 1;

}

else {

high = mid - 1;

}

}

return r\_value;

}

(A) n1=33, n2=4 (B) n1=33, n2=5 (C) n1=34, n2=4 (D) n1=34, n2=5

1. 經過運算後，右側程式的輸出為何？

for (i=1; i<=100; i=i+1) { b[i] = i;

}

a[0] = 0;

for (i=1; i<=100; i=i+1) { a[i] = b[i] + a[i-1];

}

printf ("%d\n", a[50]-a[30]);

(A) 1275

(B) 20 (C) 1000

(D) 810

1. 函數 **f** 定義如下，如果呼叫 **f(1000)**，指令**sum=sum+i** 被執行的次數最接近下列何者？

int f (int n) { int sum=0; if (n<2) {

return 0;

}

for (int i=1; i<=n; i=i+1) { sum = sum + i;

}

sum = sum + f(2\*n/3); return sum;

}

(A) 1000

(B) 3000

(C) 5000

(D) 10000

1. List 是一個陣列，裡面的元素是 element， 它的定義如右。List 中的每一個 element 利用**next** 這個整數變數來記錄下一個 element

struct element { char data; int next;

}

void RemoveNextElement (element list[], int current) {

if (list[current].next != -1) {

/\*移除 current 的下一個 element\*/

}

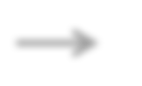
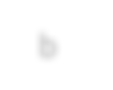
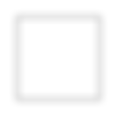
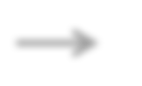
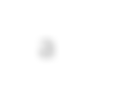
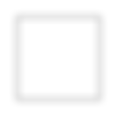
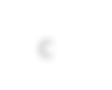
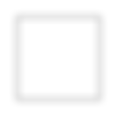
}

在陣列中的位置，如果沒有下一個 element， **next** 就會記錄-1。所有的 element 串成了一個串列 (linked list)。例如在 **list** 中有三筆資料

1 2 3

|  |  |  |
| --- | --- | --- |
| data = ‘a’  next = 2 | data = ‘b’  next = -1 | data = ‘c’  next = 1 |

它所代表的串列如下圖



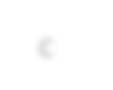
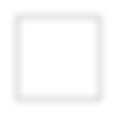
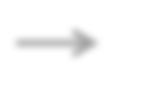
c

a

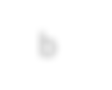
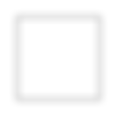
b

**RemoveNextElement** 是一個程序，用來移除串列中 **current** 所指向的下一個元素，但是必須

保持原始串列的順序。例如，若 **current** 為 3 (對應到 **list[3]**)， 呼叫完 **RemoveNextElement** 後，串列應為



c



b

請問在空格中應該填入的程式碼為何?

* 1. **list[current].next = current ;**
  2. **list[current].next = list[list[current].next].next ;**
  3. **current = list[list[current].next].next ;**
  4. **list[list[current].next].next = list[current].next ;**

1. 請問以 **a(13,15)**呼叫右側 **a()**函式，函式執行完後其回傳值為何？

int a(int n, int m) { if (n < 10) {

if (m < 10) { return n + m ;

}

else {

return a(n, m-2) + m ;

}

}

else {

return a(n-1, m) + n ;

}

}

* 1. 90

(B) 103

1. 93
2. 60
3. 一個費式數列定義第一個數為 0 第二個數為 1 之後的每個數都等於前兩個數相加，如下所示:

int a=0; int b=1;

int i, temp, N;

…

for (i=2; i<=N; i=i+1) { temp = b;

(a) ; a = temp;

printf ("%d\n", (b) );

0、1、1、2、3、5、8、13、21、34、55、89…。 右列的程式用以計算第 N 個(N≥2)費式數列的數值， 請問 (a) 與 (b) 兩個空格的敘述(statement)應該為何？

(A) (a) **f[i]=f[i-1]+f[i-2]** (b) **f[N]**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| (B) | (a) **a** | **=** | **a** | **+** | **b** | (b) **a** | } |
| (C) | (a) **b** | **=** | **a** | **+** | **b** | (b) **b** |  |
| (D) | (a) **f[i]=f[i-1]+f[i-2]** | | | | | (b) **f[i]** | |

1. 請問右側程式輸出為何？

int A[5], B[5], i, c;

…

for (i=1; i<=4; i=i+1) { if (B[i] > A[i]) {

c = c + (B[i] % A[i]);

}

else {

c = 1;

}

}

printf ("%d\n", c);

|  |  |  |
| --- | --- | --- |
| (A) | 1 | for (i=1; i<=4; i=i+1) { |
| (B) | 4 | A[i] = 2 + i\*4; |
| (C) | 3 | B[i] = i\*5; |
| (D) | 33 | } |
|  |  | c = 0; |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 10. | 給定右側 **g()** | 函式，**g(13)** | 回傳值為何？ | int g(int a) { if (a > 1) { |
| 1. 16 2. 18 3. 19 4. 22 | | | | return g(a - 2) + 3;  }  return a;  } |

1. 定義 **a[n]** 為一陣列(array)，陣列元素的指標為 0 至 n-1。若要將陣列中 **a[0]**的元素移到 **a[n-1]**，右側程式片段空白處該填入何運算式？

int i, hold, n;

…

for (i=0; i<= ; i=i+1) { hold = a[i];

a[i] = a[i+1]; a[i+1] = hold;

}

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | |  | | |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

* 1. **n+1**
  2. **n**
  3. **n-1**
  4. **n-2**

1. 給定右側函式 **f1()** 及 **f2()**。**f1(1)**運算過程中，以下敘述何者為錯？

void f1 (int m) { if (m > 3) {

printf ("%d\n", m); return;

}

else {

printf ("%d\n", m); f2(m+2);

printf ("%d\n", m);

}

}

void f2 (int n) { if (n > 3) {

printf ("%d\n", n); return;

}

else {

printf ("%d\n", n); f1(n-1);

printf ("%d\n", n);

}

}

* 1. 印出的數字最大的是 4
  2. **f1** 一共被呼叫二次
  3. **f2** 一共被呼叫三次
  4. 數字 2 被印出兩次

1. 右側程式片段擬以輾轉除法求 **i** 與 **j** 的最大公因數。請問 **while** 迴圈內容何者正確？

i = 76;

j = 48;

while ((i % j) != 0) {

\_

\_

\_

}

printf ("%d\n", j);

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| (A) | **k**  **i** | **=**  **=** | **i %**  **j;** | **j;** |
|  | **j** | **=** | **k;** |  |
| (B) | **i**  **j** | **=**  **=** | **j;**  **k;** |  |
|  | **k** | **=** | **i %** | **j;** |
| (C) | **i**  **j** | **=**  **=** | **j;**  **i %** | **k;** |
|  | **k** | **=** | **i;** |  |
| (D) | **k** | **=** | **i;** |  |

**i = j;**

**j = i % k;**

1. 右側程式輸出為何？

void foo (int i) { if (i <= 5) {

printf ("foo: %d\n", i);

}

else {

bar(i - 10);

}

}

void bar (int i) { if (i <= 10) {

printf ("bar: %d\n", i);

}

else {

foo(i - 5);

}

}

void main() { foo(15106); bar(3091); foo(6693);

}

* 1. bar: 6

bar: 1

bar: 8

* 1. bar: 6

foo: 1

bar: 3

* 1. bar: 1

foo: 1

bar: 8

* 1. bar: 6

foo: 1

foo: 3

1. 若以 **f(22)**呼叫右側 **f()**函式，總共會印出多少數字？

void f(int n) { printf ("%d\n", n); while (n != 1) {

if ((n%2)==1) {

n = 3\*n + 1;

}

else {

n = n / 2;

}

printf ("%d\n", n);

}

}

* 1. 16
  2. 22
  3. 11
  4. 15

1. 右側程式執行過後所輸出數值為何？

void main () { int count = 10;

if (count > 0) { count = 11;

}

if (count > 10) { count = 12;

if (count % 3 == 4) { count = 1;

}

else {

count = 0;

}

}

else if (count > 11) { count = 13;

}

else {

count = 14;

}

if (count) { count = 15;

}

else {

count = 16;

}

printf ("%d\n", count);

}

* 1. 11
  2. 13
  3. 15
  4. 16

1. 17.

}

else {

allBig = FALSE;

}

}

if (allBig == TRUE) {

printf ("%d is the smallest.\n", val);

}

else {

printf ("%d is not the smallest.\n", val);

}

}

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 右側程式片段主要功能為：輸入  六個整數，檢測並印出最後一個數字是否為六個數字中最小的 值。然而，這個程式是錯誤的。請問以下哪一組測試資料可以測試出程式有誤？ | | | | | | #define TRUE 1  #define FALSE 0  int d[6], val, allBig;  …  for (int i=1; i<=5; i=i+1) { scanf ("%d", &d[i]);  }  scanf ("%d", &val); | |
| (A) **11** | **12** | **13** | **14** | **15** | **3** | allBig = TRUE; | |
| (B) **11** | **12** | **13** | **14** | **25** | **20** | for (int i=1; i<=5; i=i+1) { | |
| (C) **23** | **15** | **18** | **20** | **11** | **12** | if (d[i] > val) { | |
| (D) **18** | **17** | **19** | **24** | **15** | **16** | allBig = | TRUE; |

1. 程式編譯器可以發現下列哪種錯誤?
   1. 語法錯誤
   2. 語意錯誤
   3. 邏輯錯誤
   4. 以上皆是
2. 大部分程式語言都是以列為主的方式儲存陣列。在一個 8x4 的陣列(array) **A** 裡，若每個元素需要兩單位的記憶體大小，且若 **A[0][0]**的記憶體位址為 108 (十進制表示)，則**A[1][2]**的記憶體位址為何？

(A) 120

(B) 124

(C) 128

(D) 以上皆非

1. 右側為一個計算 n 階層的函式，請問該如何修改才會得到正確的結果？
2. int fun (int n) {
3. int fac = 1;

3. if (n >= 0) {

4. fac = n \* fun(n - 1); 5. }

6. return fac; 7. }

* 1. 第 2 行，改為 **int fac = n;**
  2. 第 3 行，改為 **if (n > 0) {**
  3. 第 4 行，改為 **fac = n \* fun(n+1);**
  4. 第 4 行，改為 **fac = fac \* fun(n-1);**

1. 右側程式碼，執行時的輸出為何？

void main() {

for (int i=0; i<=10; i=i+1) { printf ("%d ", i);

i = i + 1;

}

printf ("\n");

}

(A) **0 2 4 6 8 10**

(B) **0 1 2 3 4 5 6 7 8 9 10**

(C) **0 1 3 5 7 9**

(D) **0 1 3 5 7 9 11**

1. 22.

|  |  |  |
| --- | --- | --- |
| 右側 **f()**函式執行後所回傳的值為何？ | | int f() { int p = 2; |
| (A) | 1023 | while (p < 2000) { |
| (B) | 1024 | p = 2 \* p; |
| (C) | 2047 | } |
| (D) | 2048 | return p; |
|  | | } |

1. 右側 **f()**函式 (a), (b), (c) 處需分別填入哪些數字，方能使得 **f(4)** 輸出 2468 的結果？

int f(int n) { int p = 0; int i = n;

while (i >= (a) ) {

p = 10 – (b) \* i; printf ("%d", p);

i = i - (c) ;

}

}

(A) **1, 2, 1**

(B) **0, 1, 2**

(C) **0, 2, 1**

(D) **1, 1, 1**

1. 右側 **g(4)**函式呼叫執行後，回傳值為何？

int f (int n) { if (n > 3) {

return 1;

}

else if (n == 2) { return (3 + f(n+1));

}

else {

return (1 + f(n+1));

}

}

int g(int n) { int j = 0;

for (int i=1; i<=n-1; i=i+1) { j = j + f(i);

}

return j;

}

* 1. 6
  2. 11
  3. 13
  4. 14

1. 右側 **Mystery()**函式 **else** 部分運算式應為何，才能使得 **Mystery(9)** 的回傳值為 **34**。

int Mystery (int x) { if (x <= 1) {

return x;

}

else {

return ;

}

}

* 1. **x + Mystery(x-1)**
  2. **x \* Mystery(x-1)**
  3. **Mystery(x-2) + Mystery(x+2)**
  4. **Mystery(x-2) + Mystery(x-1)**