

32 位机上根据下面的代码，问哪些说法是正确的？

```
signed char a = 0xe0;
```

```
unsigned int b = a;
```

```
unsigned char c = a;
```

A. `a>0 && c>0` 为真 B. `a == c` 为真 C. `b` 的十六进制表示是：0xfffffe0 D. 上面都不对

**分析：**坑爹丫，有木有！10 个人 9 个这个恐怕都不敢确定！（敢肯定的要么是高手，要么就是错的！）B me 认为是错的，一个 `uchar` 和一个 `schar` 比较，真的就是一个字节在比较吗？C me 认为是对的，将一个 `schar` 赋值给一个 `uint`，要不要符号扩展呢？是绝对会还是可能会呢？细节到底是神马？O\_\_O"...A 貌似比较确定，肯定是错的，肯定？

**揭露真相：**A 确实是错的，B 也是错的，C 是对的，所以 D 也是错的。理由？A 错是因为，`a` 是负数，`c` 是正数，即使跟 0 比较要转换到 `int`（后面暂不区分转换和类型提升，总之就是类型变了），也是一负一正，所以 A 错。B 呢？是说一正一负不会相等，难道是因为这吗？难道不是吗？首先说 `a` 和 `c` 的二进制表示一模一样，都是 0xe0，那么比较就不相等？！是的，比较的时候全部转换为 `int`，所以呢，`a` 还是作为一个负数存在，`c` 作为一个正数存在，于是不相等了，所以 B 错。C 肿么就对了？`a` 是一个 `schar`，赋值给 `uint` 的 `b`，前若干个字节不是补 0 吗？首先 `schar` 转换为 `int`，然后 `int` 转换成 `uint`，所以最初是符号扩展，然后一个 `int` 赋值给了 `uint`，C correct！me 曾经要写一篇关于 `c` 的类型以及指针的 blog，不过最后没有完成，不过还是可以参考一下的。

下面哪些选项能编译通过？

```
int i;
```

```
char a[10];
```

```
string f();
```

```
string g(string &str);
```

A. `if(!i){f();}` B. `g(f());` C. `a=a+1;` D. `g("abc");`

**分析：**再次坑爹有木有！（其实 me 比较确信这道题，是坑别人的爹，O\_\_O"...）A 绝对是正确的，C 绝对是错的，D 基本肯定是错的，那 B 呢？要么 error，要嘛 warning！如果是 warning 但是没有 error，这算神马情况呢？B 确实不应该选，至少语义上不该选！`f()` 返回一个临时量，然后传给 `g`

函数，而 `g` 的参数是非 `const` 引用，是不能引用临时量的！为嘛，如果 `g` 中修改了传进来的 `string`，那么会是怎么一回事呢？修改了一个临时量的值？那这意义何在呢？但是如果将 `g` 的原型修改为 `string g(const string&);` 就是可以的，为什么可以？访问(只读)临时量就是正确的？那必须的，比如 `u` 可能想知道 `a+b` 的结果是多少，然后输出！`a+b` 的结果就是一个临时量。如果说修改 `a+b` 的结果，这是神马个逻辑？！

**真相：** `C` 错是以为 `a` 是一个地址常量，不可能再被赋值。`D` 为嘛错呢？"`abc`" 的类型可是 `const char*` 呢，是个常量指针呢！（可以用来初始化 `string`。）

`int a[10];` 问下面哪些不可以表示 `a[1]` 的地址？

A. `a+sizeof(int)` B. `&a[0]+1` C. `(int*)&a+1` D. `(int*)((char*)&a+sizeof(int))`

分析：奇葩丫！（其实并不奇葩！）腾讯的题目有时候出的的确有水平丫，虽然出的太有水平了分就考不高了。`me` 想哭丫，想来想去还是在 `A` 和 `B` 中选错了，`%>_<%`，当时还特意提醒自己来着的，`O__O`“...`c++` 中的 `sort` 如何用来排序 `a` 数组呢？`sort(a, a+N);` 或是 `sort(a, a+sizeof(a)/sizeof(a[0]));` 当时懵了，实际上 `a+1`，就是 `a[1]` 的地址呢！`a` 的类型是 `int[10]`，`a+1` 和一个 `int*` 类型的 `+1` 效果一样，都表示偏移 1 个元素，所以 `A` 不能表示。（选错误的！）`C` 能表示是因为取了首地址作为一个 `int*` 然后 `+1`，就是偏移一个元素，所以不选。`B` 肿么说呢，`me` 一直一位 `&a[0]` 是一个普普通通的地址，`+1` 就是 `+1` 个字节，实际上是 `+1` 个元素！`D` 也能表示？将 `a` 的首地址转换为一个 `char*` 指针，这个时候 `+1` 是偏移一个 `char`，也就是一个字节，实际上应该偏移 `sizeof(int)` 个字节才能到达 `a[1]`，所以 `D` 可以表示（不选）。不多说了。（如果是二维数组是不是会更懵呢，`O__O`“...）

问下面的数据都存放在哪些存储区？

```
int main()
{
    char *p = "hello,world";
    return 0;
}
```

A. ... B. ... C. 栈和常量区 D. 栈和堆

**分析：** "`hello,world`" 是常量，赶脚应该就是 `C` 吧，应该大家感觉都一样。这里不涉及什么堆的事。

假设在一个 32 位 little endian 的机器上运行下面的程序，结果是多少？

```
#include <stdio.h>

int main()
{
    long long a = 1, b = 2, c = 3;
    printf("%d %d %d\n", a, b, c);
    return 0;
}
```

A. ... B. ... C. ... D. ...

**分析：**貌似问题没有想的那么简单。如果说运行结果，很简单，有人是 1 0 2 (VC6.0 和 VC2008)；有人是 1 2 3。涉及到 little/big endian 和参数入栈的问题，me 表示现在有点无能为力，O\_\_O"...

Address	0 - 3	4 - 7	8 - B	C - F
BFFFF170	01000000	34F2FFBF	01000000	00000000
BFFFF180	02000000	00000000	03000000	00000000
BFFFF190	90840408	00000000	00000000	D3F4E2B7
BFFFF1A0	01000000	34F2FFBF	3CF2FFBF	70C9FDB7

在 32 和 64 上面，long long 都是 8 字节，printf("%d %d %d\n", a, b, c);会依次从 a 的地址开始输出 3 个整型数据（4B）一共是 12B，调用 printf 时，函数参数的压栈顺序是 c, b, a 且地址是连续存放的，小端情况下从 a 开始的栈去内存内容如下：

```
0x 01 00 00 00 00 00 00 00
0x 02 00 00 00 00 00 00 00
0x 03 00 00 00 00 00 00 00
```

所以连续输出 12 个字节的結果就是：1 0 2

## 【分析】：

传入参数，由右往左，栈空间内存从高往低，little endian, 栈空间如下：

内存高位->

00000000 00000011

00000000 [00000010](c)

[00000000](b) [00000001](a)

<-内存低位

因此 printf 会按照 4bytes 取参数.

## 【答案】

输出: 1, 0, 2

下面哪些函数调用必须进入内核才能完成？

A. fopen B. exit C. memcpy D. strlen

**分析：**有些无能为力。A 是要打开文件的，貌似设计很多内核操作丫；exit 是退出进程，结束进程，应该也要深入内核。memcpy，me 一直犹豫用户区的数据拷贝要不要通过内核。strlen me 感觉关系不大。

内存管理中的 LRU 方法是用来管理神马的？

A. 虚拟内存的分配 B. 虚拟内存的释放 C. 物理内存的分配 D. 物理内存的释放

**分析：**貌似是用来关系物理块的，后面的填空题正好有说，O\_\_O"...

关于 DMA 的说法，哪些是错误的？

A. DMA, Direct Memory Access 直接存储器访问，使得不同的速度的硬件设备可以直接通信，不通过 CPU 干预；

B. DMA 访问的时候需要从 CPU 那里夺得总线控制权，然后...

C. DMA 速度快；

D. DMA 不需要中断控制，CPU 管理不要它；

死锁发生的必要条件？

A. 互斥条件 B. 请求和保持 C. 不可剥夺 D. 循环等待

**分析：**ABCD 就是死锁的四个必要条件，操作系统书上貌似说的很明确。

有两个线程，最初 n=0，一个线程执行 n++; n++; 另一个执行 n+=2; 问，最后可能的 n 值？



A. 1 B. 2 C. 3 D. 4

**分析：**D 顺序执行以下，就可以。B 的话，让后面一个执行到+2，但不要写结果，然后前一个执行完，然后写结果，为 2。C 3 的话，也好分析。A 不可能！肿么可能呢？肿么可能结果只为 1 呢？两个线程都会 +2，+1 何从谈起？先 +1，然后让后面的加法错了，然后结果写进去？前一个 ++n 都没执行的话，后一个又肿么会执行呢？总之不可能是 1！不可能！O\_\_O"…(坚决不相信它可以。)

下面哪些说法正确？

A. 数组和链表都可以随机访问

B. 数组的插入和删除可以  $O(1)$

C. 哈希表么法范围检查

D. ...

**分析：**总之 ABD 给人的感觉是显而易见的错丫，有木有，所以排除法还是能用的！至于 hash 结构，确实也不可以范围检查，因为 key 映射为 value，完全将根据的 hash 函数，而这个函数一般不满足原来的单调性，实际上就应该满足！因为 hash 函数的设计是要 value 的映射随机、均匀！

基于比较的排序的时间复杂度下限是多少？

A.  $O(n)$  B.  $O(n^2)$  C.  $O(n \log n)$  D. ...

**分析：**貌似是数据结构上面的一个结论，基于比较的排序的时间复杂度不能比  $O(n \log n)$  地。

下面图的拓扑排序可能是？

A...

**分析：**对于知道拓扑排序的，应该很容易作答(me 有时候在怀疑自己理解的是否正确？！)。

求 n 个数中的最大值和最小值，最少的比较次数是？

A.  $4n/3$  B.  $2n-2$  C.  $n-1$  D.  $3n/2$

**分析：**虽然 me 不知道很高深的算法，但是 me 想丫，如果是 2 1 3 4 5 6 这样的序列，u 认为最少是多少次呢？me 感觉是  $n-1$ 。但是，题目也许是另外一个意思，也就是对于普通的序列，求最大值和最小值，能将比较次数降到多少？(me 貌似见到过一些方法，但是忘了，O\_\_O"…)

一棵二叉树的先序遍历是 f b a c d e g h，中序遍历是 a b d c e f g h，问后序遍历是神马？

A. ... B. ... C. ... D. ...

**分析：**构建二叉树，然后看看后序遍历是神马？a d e c b h g f, O\_\_O"…，突然感觉一不小心 g f 和 f g 就写反了，me 应该没有吧？！

网卡、交换机、路由器和 UDP 分别工作网络的哪些层？

A. ...

B. ...

C. ...

D. ....

**分析：**值根据 UDP 在传输层，me 就选出答案了：物理层、数据链路层、网络层和传输层。

子网掩码 25..255.255.224 可以将网络 x.x.x.x 分成多少个子网？

A. ... B. ... C. 8 D. 32

**分析：** $224 = 128 + 64 + 32 = 1110\ 0000\ B$ ，me 一看，后面 5 个 0，就是  $2^5 = 32$  吧？

TCP 协议栈的定时器有哪些？

A. ... B. ... C. ... D. ...

**分析：**不会的路过。

高内聚和低耦合，下面哪个耦合度最高？

A. 通过函数参数传递...

B. 一个函数修改另外一个函数中的数据；

C. 通过全局变量...

D. 通过指示器...

**分析：**一看全局变量，就是它无疑了。O\_\_O"…

关于访问，下面哪些是后台绝对不会执行的？

A. 本地查查 DNS，是否有 sinaapp.com 的 ip；

B. 通过 cn. 查找 ip；

C. 通过 com. 查找 ip；

D. 浏览器发送 http get 请求；

D. 服务器回送 xxx.jpg;

分析：蒙也是蒙 B，O\_\_O"...

在一个 big endian 的 32 位的计算机上，b 的结果是？(该处 1 个空)

```
unsigned int a = 0x1234;
```

```
char b = *((char*)&a);
```

分析：想不到又一个 big/little endian 的问题。这个结果，me 还比较确定写，就是 0。uint 是 4 个字节，0x1234 的完整表示是 0x 00 00 12 34，因为是 big endian，所以，所以 &(取地址) 的话，如果当字节看的话，取到了最左边 00 字节的地址，一定要转换成 char 看的话，值就是 0。

一个有 800 个结点的完全二叉树，问有多少个叶子结点？(该处 1 个空)

虽然 me 忘记是神马公式计算了，只感觉和  $n/2$  有关系。然后随便画几个试试，就可以找出来  $(n+1)/2$  的规律来，所以 400。

下面 get 是求一个二维数组元素的函数，请补全。(1 个空)

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

```
#include <stdlib.h>
```

```
#define M 3
```

```
#define N 4
```

```
int get(int *a, int i, int j)
```

```
{
```

```
    int v;
```

```
    if(a == NULL || i < 0 || i >= M || j < 0 || j >= N) exit(1); //
```

```
    v = *(a+i*N+j); // 这里有一个空
```

```
return v;
}
int main()
{
    int a[M[N = {{1,2,3,4},{5,6,7,8},{9,10,11,12}}];
    int v;
    v = get(a, 2, 1);
    printf("a[2][1] == %d\n", v);
    return 0;
}
```

**分析：**差点写错了，还好迷途知返了，o\_o"...

补全插入排序：(有 2 个空)

```
#include <stdio.h>
#include <stdlib.h>
int insert_sort(int *p, int count)
{
    int i, j, tmp;
    if(p == NULL || count < 0) return 0; //
    for(i=1; i<count; i++){
        tmp = p[i];
        j = i-1;
        while(j>=0 && p[j]>tmp){ // 此处判断条件一个空
```



```

    p[j+1] = p[j];
    --j;
}
p[j+1] = tmp; // 此处一个空
}
return 1;
}
int main()
{
    int i, a[10] = {3, 2, 1, 7, 8, 10, 4, 5, 6, 9};
    insert_sort(a, 10);
    for(i=0; i<10; i++)
        printf("%d ", a[i]);
    printf("\n");
    return 0;
}

```

**分析：**me 感觉 me 的代码还是比较工整的，肿么看，...

使用 FIFO 管理页面请求，如果分配的物理块  $M=3$  或是 4，请求顺序如下：4 3 2 4 4 3 5 4 5 3 1 5 1 5 1 5 4，问两种情况下页面失效的次数各是多少？(2 个空)

**分析：**7 和 7，这是 me 的结果。

一个网络图，问 Q 路由器到某个网络要进行跳转的下一个 ip 是多少？(有 1 个空)

**分析：**me 一下子就犹豫了，貌似赶脚也不是正确答案，就不多说了。

软件可靠性评估的两个指标是神马？(2 个空)

**分析：**不会的路过。