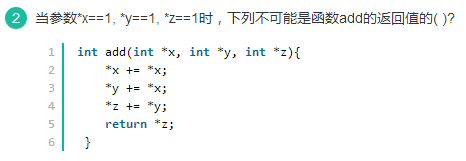
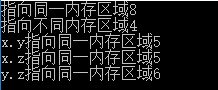
# 2016/9/17

## #



x,y,z是否指向同一地址（或者说调用该函数时是否实参相同）：

|  |
| --- |
| **#include<iostream>**  **#include<stdio.h>**  **using namespace std;**  **int add(int \*x, int \*y, int \*z)**  **{**  **\*x += \*x;**  **\*y += \*x;**  **\*z += \*y;**  **return \*z;**  **}**    **int main()**  **{**  **//划分三块内存区域**  **int a=1,b=1,c=1;**  **//注意经过一次add()之后，a，b，c的值有可能发生改变，注意要值要归1**  **printf("指向同一内存区域%d\n",add(&a,&a,&a));**  **a=1,b=1,c=1;**  **printf("指向不同内存区域%d\n",add(&a,&b,&c));**  **a=1,b=1,c=1;**  **printf("x.y指向同一内存区域%d\n",add(&a,&a,&b));**  **a=1,b=1,c=1;**  **printf("x.z指向同一内存区域%d\n",add(&a,&c,&a));**  **a=1,b=1,c=1;**  **printf("y.z指向同一内存区域%d\n",add(&a,&c,&c));**  **return 0;**  **}** |



## #

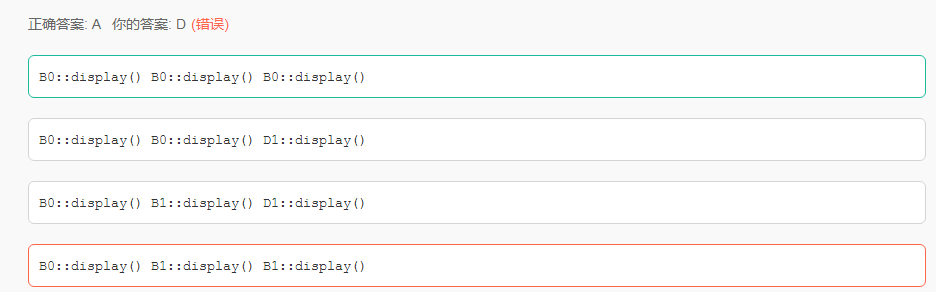


不可以，void类型没有分配内存，而引用必须是另一个固定内存变量的别名，所以不能指向void

## #

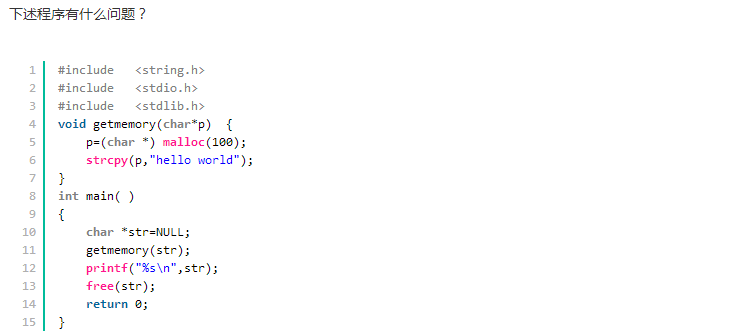


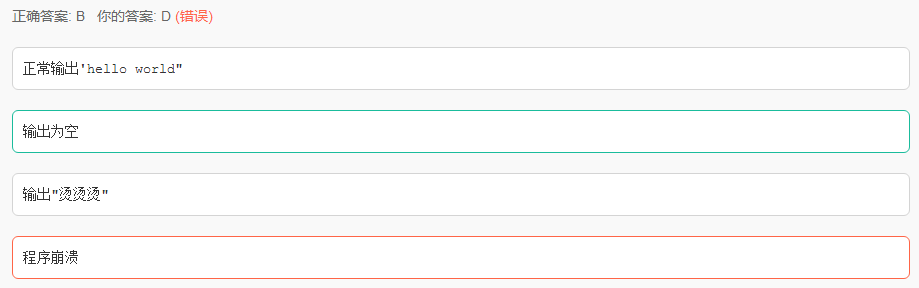


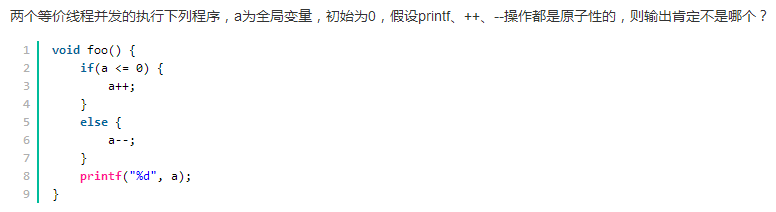












A:01 B:10 C:12 D:22

对于B选项：P1执行程序，输入1，P2执行程序，输出0；

对于C选项：初始为0，P1执行完判断语句，决定要执行a++，中断，P2进行判断，此时a仍然等于0，执行判断语句，并执行输入，得到1，P1然后继续执行，此时它该执行a++，这时a=1，执行并输出，结果为2；

对于D答案：初始为0，P1执行完判断语句，决定要执行a++，中断，P2进行判断，此时a仍然等于0，执行a++，得到a=1，中断，P1继续执行a++，a=2，P1输出，得到2，P1结束，P2继续执行输出语句，得到2；

# 2016/9/18

## #



## #

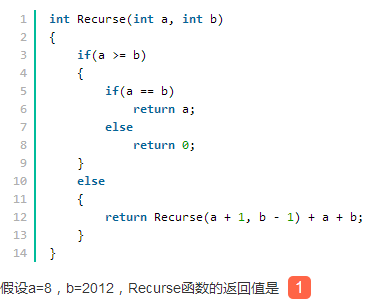
下列给定程序中，函数fun的功能是：把形参a所指数组中的最大值放在a[0]中，接着求出a所指数组中的最小值放在a[1]中，再把a所指数组元素中的次大值放在a[2]中，把a数组元素中的次小值放在a[3]中，依此类推。  
例如，若a所指数组中的数据最初排列为：1、4、2、3、9、6、5、8、7，按规则移动后，数据排列为：9、1、8、2、7、3、6、4、5.形参n中存放a所指数组中数据的个数。   
请在程序的下画线处填入正确的内容并将下画线删除，使程序得出正确的结果。 试题程序：

#define N 9

/\*\*\*\*\*\*\*\*\*\*found\*\*\*\*\*\*\*\*\*\*/  
void fun(int a  , int n)  
{  
int i, j, max, min, px, pn, t;  
/\*\*\*\*\*\*\*\*\*\*found\*\*\*\*\*\*\*\*\*\*/  
for (i = 0; i < n - 1; i +=  )  
{  
max = min = a[i];  
px = pn = i;  
for (j =  ; j < n  ; j ++)  
{  
  
if (max < a[j])  
{  
max = a[j];  
px = j;  
}  
if (min > a[j])  
{  
min = a[j];  
pn = j;  
}  
}  
if (px != i)  
{  
t = a[i]; a[i] = max; a[px] = t;  
if (pn == i) pn = px;  
}  
if (pn != i + 1)  
{  
t = a[i + 1];  
a[i + 1] = min;  
a[pn] = t;  
}  
}  
}

[],2,i+1

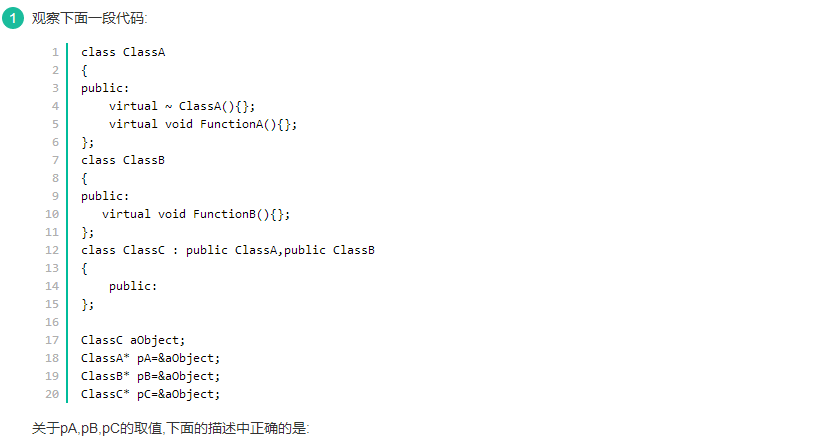
## #



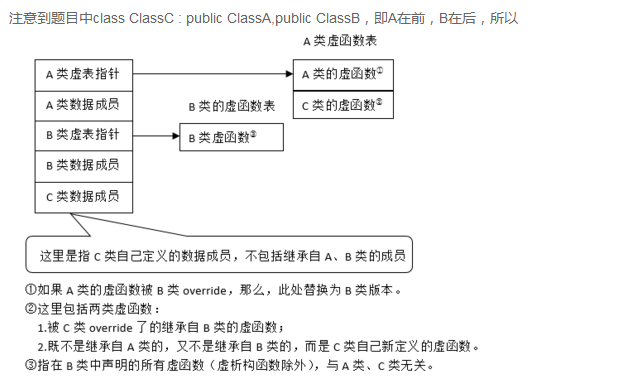
参考答案 2025050

# 2016/9/19

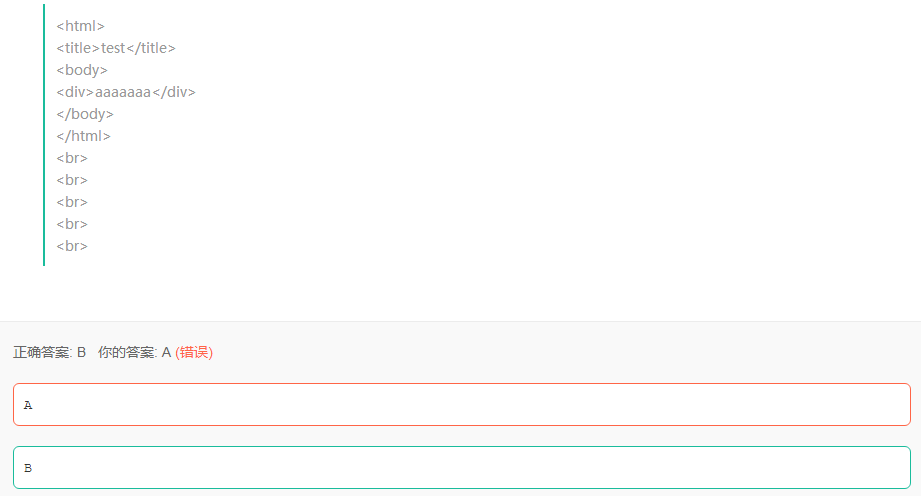
## #







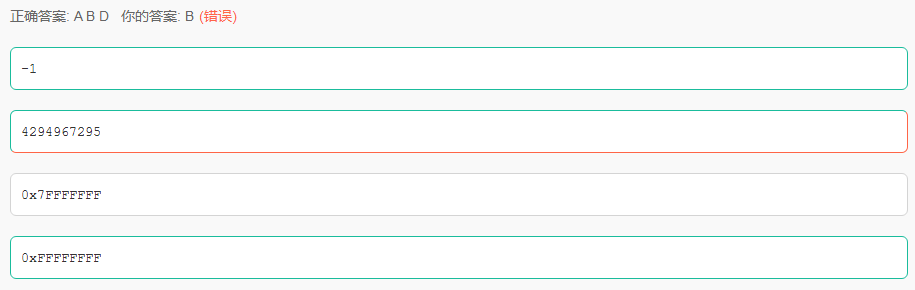


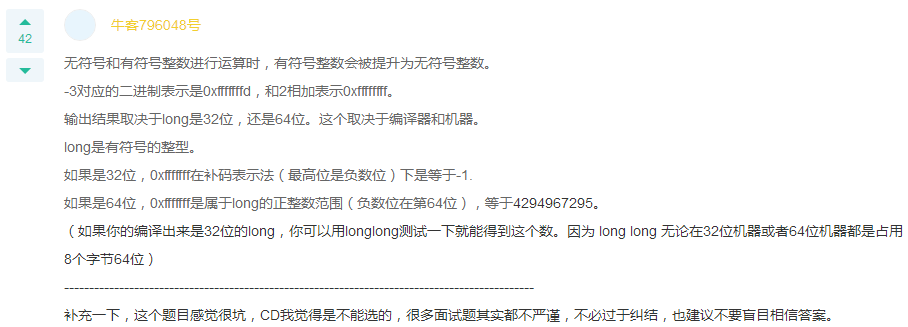


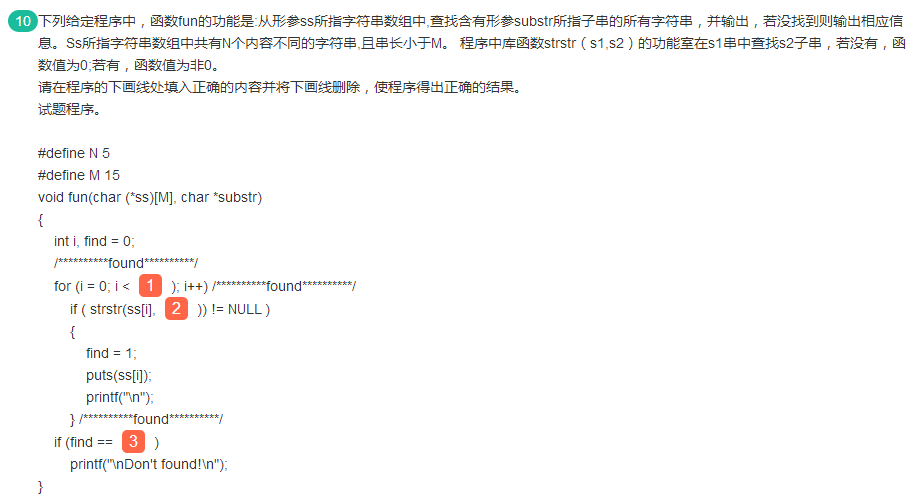


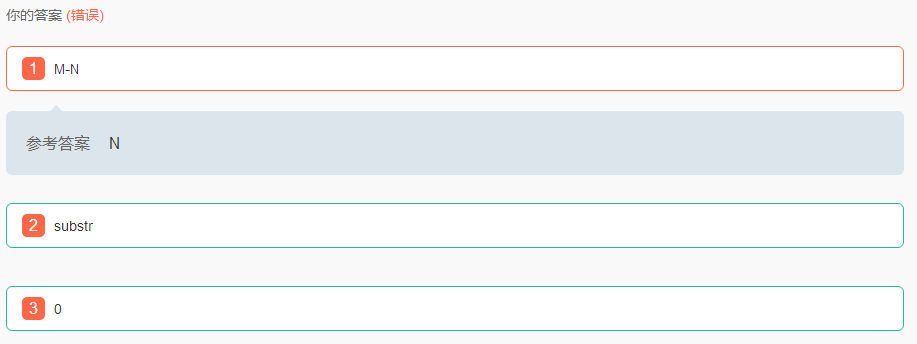
把&&左边和右边位置调换一下就对了





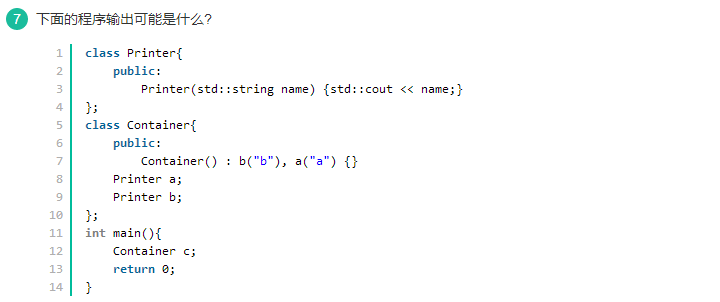


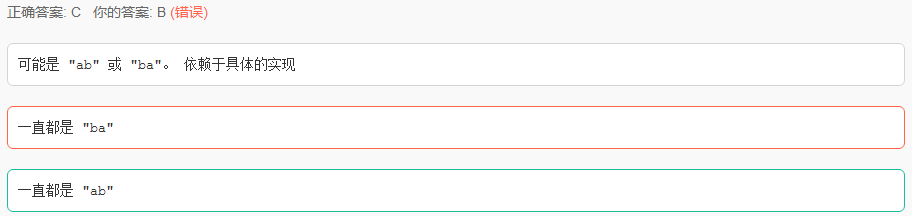




# 2016/9/20

## #





答案C:一直都是ab。  
派生类实例化时，先调用基类的构造函数，然后是派生类的类成员变量构造函数（构造的顺序是按照成员变量的定义先后顺序，而不是按照初始化列表的顺序），最后是派生类的构造函数。  
程序中，先定义了 Printer a;后定义了Printer b;所以输出一定是ab。

## #



37. 对于只做输入的参数：

a) 始终用const限制所有指向只输入参数的指针和引用。

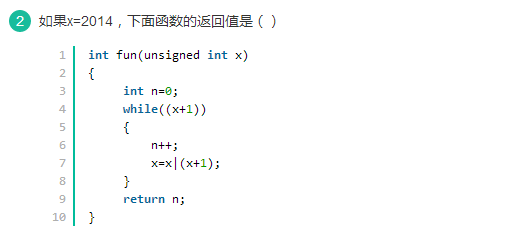
b) 优先通过值来取得原始类型和复制开销比较低的值的对象。

c) 优先按const的引用取得其他用户定义类型的输入。

d) 如果函数需要其参数的副本，则可以考虑通过值传递代替通过引用传递。这在概念上等同于通过const引用传递加上一次复制，能够帮助编译器更好的优化掉临时变量。

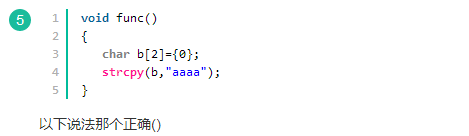
# 2016/9/23

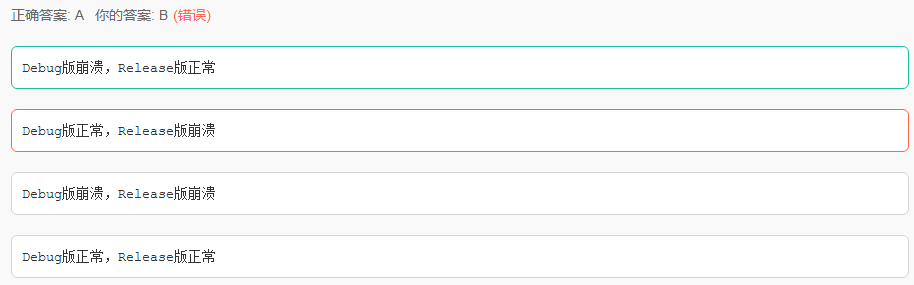
## #



x&(x-1)统计1的个数，x|(x+1)统计0的个数

## #

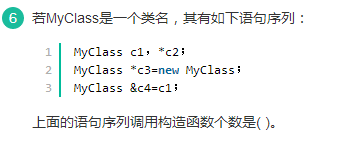




因为在Debug中有ASSERT断言保护，所以要崩溃，而在Release优化中就会删掉ASSERT,所以会出现正常运行。

但是不推荐如此做，因为这样会覆盖不属于自己的内存，这是搭上了程序崩溃的列车，即未定义行为，出现什么后果都有可能的，程序员的职责是保证不越界，而不是追问越界之后会发生什么。就像生活中违法不一定立刻被抓，但是迟早是要被抓的。

## #



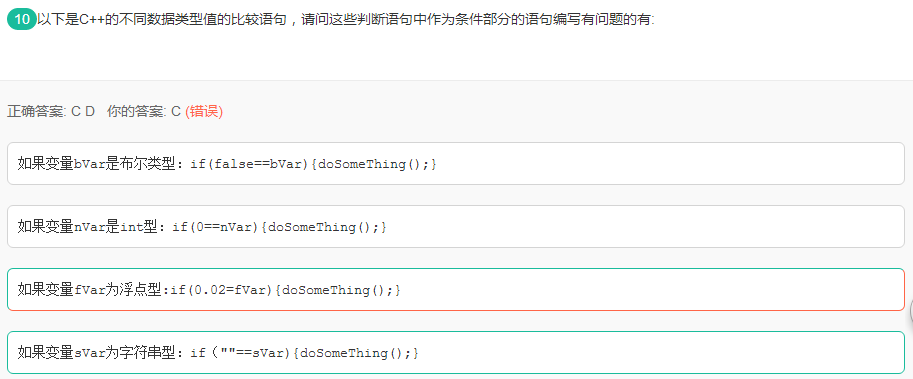
2

只有c1和c3会调用构造函数。

c2只是一个指针，用来动态描述对象的，不需要类的定义。

c4只是一个c1的别名，说白了c4就是c1  不占空间

## #

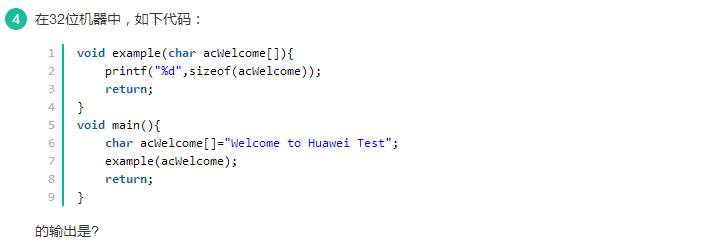


c选项，float有精度问题，尾数不精确，比较会出问题

d选项，字符串比较，一般用strcmp（str1，str2），直接用==比较两个字符串，应该比较的是字符串的首地址是否相等，那就不是真正的字符串比较了

# 2016/10/9

## #



这题的正确答案应该是32位系统是4 64位系统是8，数组作为函数的参数是会退化为函数指针的，想想看，数组作为函数参数的时候经常是需要传递数组大小的。

## #



最小：100000(-32)；

最大：011111(31)

## #



答案：

a.成员函数被重载的特征：

（1）相同的范围（在同一个类中）；

（2）函数名字相同；

（3）参数不同；

（4）virtual 关键字可有可无。

b.覆盖是指派生类函数覆盖基类函数，特征是：

（1）不同的范围（分别位于派生类与基类）；

（2）函数名字相同；

（3）参数相同；

（4）基类函数必须有virtual 关键字。

c.“隐藏”是指派生类的函数屏蔽了与其同名的基类函数，规则如下：

（1）如果派生类的函数与基类的函数同名，但是参数不同。此时，不论有无virtual关键字，基类的函数将被隐藏（注意别与重载混淆）。

（2）如果派生类的函数与基类的函数同名，并且参数也相同，但是基类函数没有virtual 关键字。此时，基类的函数被隐藏（注意别与覆盖混淆）