1、10、10、4、4四个数，怎么算出24点？  
(10\*10-4)/4=24  
2、下列表达式在32位机器编译环境下的值（）

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/hackbuteer1/article/details/8476206)

1. class A
2. {
3. };
5. class B
6. {
7. public:
8. B();
9. virtual ~B();
10. };
12. class C
13. {
14. private:
15. #pragma pack(4)
16. **int** i;
17. **short** j;
18. **float** k;
19. **char** l[64];
20. **long** m;
21. **char** \*p;
22. #pragma pack()
23. };
25. class D
26. {
27. private:
28. #pragma pack(1)
29. **int** i;
30. **short** j;
31. **float** k;
32. **char** l[64];
33. **long** m;
34. **char** \*p;
35. #pragma pack()
36. };
38. **int** main(void)
39. {
40. printf("%d\n",sizeof(A));
41. printf("%d\n",sizeof(B));
42. printf("%d\n",sizeof(C));
43. printf("%d\n",sizeof(D));
44. return 0;
45. }

class A

{

};

class B

{

public:

B();

virtual ~B();

};

class C

{

private:

#pragma pack(4)

int i;

short j;

float k;

char l[64];

long m;

char \*p;

#pragma pack()

};

class D

{

private:

#pragma pack(1)

int i;

short j;

float k;

char l[64];

long m;

char \*p;

#pragma pack()

};

int main(void)

{

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

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

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

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

return 0;

}

A、1、4、84、82      B、4、4、82、84      C、4、4、84、82      D、1、4、82、82  
3、以下程序在32位机器下运行的结果是（）

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/hackbuteer1/article/details/8476206)

1. #pragma pack(4)
2. struct info\_t
3. {
4. unsigned **char** version;
5. unsigned **char** padding;
6. unsigned **char** extension;
7. unsigned **char** count;
8. unsigned **char** marker;
9. unsigned **char** payload;
10. unsigned **short** sequence;
11. unsigned **int** timestamp;
12. unsigned **int** ssrc;
13. };
15. union info\_u
16. {
17. unsigned **char** version;
18. unsigned **char** padding;
19. unsigned **char** extension;
20. unsigned **char** count;
21. unsigned **char** marker;
22. unsigned **char** payload;
23. unsigned **short** sequence;
24. unsigned **int** timestamp;
25. unsigned **int** ssrc;
26. };
27. #pragma pack()
29. **int** main(void)
30. {
31. printf("%d\n",sizeof(info\_t));
32. printf("%d\n",sizeof(info\_u));
33. return 0;
34. }

#pragma pack(4)

struct info\_t

{

unsigned char version;

unsigned char padding;

unsigned char extension;

unsigned char count;

unsigned char marker;

unsigned char payload;

unsigned short sequence;

unsigned int timestamp;

unsigned int ssrc;

};

union info\_u

{

unsigned char version;

unsigned char padding;

unsigned char extension;

unsigned char count;

unsigned char marker;

unsigned char payload;

unsigned short sequence;

unsigned int timestamp;

unsigned int ssrc;

};

#pragma pack()

int main(void)

{

printf("%d\n",sizeof(info\_t));

printf("%d\n",sizeof(info\_u));

return 0;

}

A、12  12      B、12  4       C、16  4   D、16  12     E、16  1  
4、以下表达式result的值是（）

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/hackbuteer1/article/details/8476206)

1. #define VAL1(a,b) a\*b
2. #define VAL2(a,b) a/b--
3. #define VAL3(a,b) ++a%b
5. **int** a = 1;
6. **int** b = 2;
7. **int** c = 3;
8. **int** d = 3;
9. **int** e = 5;
11. **int** result = VAL2(a,b)/VAL1(e,b)+VAL3(c,d);

#define VAL1(a,b) a\*b

#define VAL2(a,b) a/b--

#define VAL3(a,b) ++a%b

int a = 1;

int b = 2;

int c = 3;

int d = 3;

int e = 5;

int result = VAL2(a,b)/VAL1(e,b)+VAL3(c,d);

A、-2     B、1     C、0     D、2  
5、请写出以下程序的输出（5分）

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/hackbuteer1/article/details/8476206)

1. void swap\_1(**int** a , **int** b)
2. {
3. **int** c;
4. c = a;
5. a = b;
6. b = c;
7. return ;
8. }
9. void swap\_2(**int** &a , **int** &b)
10. {
11. **int** c;
12. c = a;
13. a = b;
14. b = c;
15. return ;
16. }
17. void swap\_3(**int** \*a , **int** \*b)
18. {
19. **int** c;
20. c = \*a;
21. \*a = \*b;
22. \*b = c;
23. return ;
24. }
26. **int** main(void)
27. {
28. **int** a = 100;
29. **int** b = 200;
30. swap\_1(a , b);
31. printf("a = %d , b = %d\n",a , b);
32. swap\_2(a , b);
33. printf("a = %d , b = %d\n",a , b);
34. swap\_3(&a , &b);
35. printf("a = %d , b = %d\n",a , b);
36. return 0;
37. }

void swap\_1(int a , int b)

{

int c;

c = a;

a = b;

b = c;

return ;

}

void swap\_2(int &a , int &b)

{

int c;

c = a;

a = b;

b = c;

return ;

}

void swap\_3(int \*a , int \*b)

{

int c;

c = \*a;

\*a = \*b;

\*b = c;

return ;

}

int main(void)

{

int a = 100;

int b = 200;

swap\_1(a , b);

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

swap\_2(a , b);

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

swap\_3(&a , &b);

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

return 0;

}

输出结果：  
a = 100 , b = 200  
a = 200 , b = 100  
a = 100 , b = 200  
6、下面的程序是否有问题，如有问题，请重构代码（5分）

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/hackbuteer1/article/details/8476206)

1. void test\_type(**bool** b , const **char** \*p , **float** f)
2. {
3. if(!b)
4. {
5. return ;
6. }
7. else if(!p)
8. {
9. return ;
10. }
11. else if(!f)
12. {
13. return ;
14. }
15. }

void test\_type(bool b , const char \*p , float f)

{

if(!b)

{

return ;

}

else if(!p)

{

return ;

}

else if(!f)

{

return ;

}

}

修改如下：

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/hackbuteer1/article/details/8476206)

1. void test\_type(**bool** b , const **char** \*p , **float** f)
2. {
3. if(!b)
4. {
5. return ;
6. }
7. else if(!p)
8. {
9. return ;
10. }
11. else if(f > -1e-10 && f < 1e-10)
12. {
13. return ;
14. }
15. }

void test\_type(bool b , const char \*p , float f)

{

if(!b)

{

return ;

}

else if(!p)

{

return ;

}

else if(f > -1e-10 && f < 1e-10)

{

return ;

}

}

7、请指出以下程序有什么问题（5分）

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/hackbuteer1/article/details/8476206)

1. void test\_mem()
2. {
3. **char** \*p = new **char**[64];
4. delete p;
5. p = NULL;
6. return ;
7. }

void test\_mem()

{

char \*p = new char[64];

delete p;

p = NULL;

return ;

}

应该修改为 delete[]p;  p指向的是一个字符型的数组空间，原来的代码只是简单的释放了指向申请空间的指针，并没有释放申请的空间，容易造成内存崩溃。  
回收用 new 分配的单个对象的内存空间的时候用 delete，回收用 new[] 分配的一组对象的内存空间的时候用 delete[]。  
8、以下程序有什么问题，请指出。

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/hackbuteer1/article/details/8476206)

1. **char**\* GetMem()
2. {
3. **char** p[] = "hello";
4. return p;
5. }
7. void test\_get\_mem()
8. {
9. **char** \*p = GetMem();
10. printf(p);
11. return ;
12. }

char\* GetMem()

{

char p[] = "hello";

return p;

}

void test\_get\_mem()

{

char \*p = GetMem();

printf(p);

return ;

}

GetMem函数中的p是一个在栈上的局部变量，当函数运行结束的时候，栈上的内容会自动释放的，此处返回的值有可能会成为一个野指针，会出现一个意想不到的结果。  
9、请写出strcpy 和 memcpy 的区别（5分）  
答：strcpy和memcpy都是标准C库函数，它们有下面的特点。  
strcpy提供了字符串的复制。即strcpy只用于字符串复制，并且它不仅复制字符串内容之外，还会复制字符串的结束符。  
strcpy函数的原型是：char\* strcpy(char\* dest, const char\* src);  
memcpy提供了一般内存的复制。即memcpy对于需要复制的内容没有限制，因此用途更广。  
memcpy函数的原型是：void \*memcpy( void \*dest, const void \*src, size\_t count );  
strcpy和memcpy主要有以下3方面的区别。  
1、复制的内容不同。strcpy只能复制字符串，而memcpy可以复制任意内容，例如字符数组、整型、结构体、类等。  
2、复制的方法不同。strcpy不需要指定长度，它遇到被复制字符的串结束符"\0"才结束，所以容易溢出。memcpy则是根据其第3个参数决定复制的长度。  
3、用途不同。通常在复制字符串时用strcpy，而需要复制其他类型数据时则一般用memcpy。  
  
10、请写出以下程序的输出结果

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/hackbuteer1/article/details/8476206)

1. class Base
2. {
3. public:
4. Base()
5. {
6. printf("I am Base()\n");
7. }
8. virtual ~Base()
9. {
10. printf("I am ~Base()\n");
11. }
12. public:
13. virtual void SayHello()
14. {
15. printf("Hello Base\n");
16. }
17. void SayWorld()
18. {
19. printf("World Base\n");
20. }
21. };
22. class Derived : public Base
23. {
24. public:
25. Derived()
26. {
27. printf("I am Derived()\n");
28. }
29. virtual ~Derived()
30. {
31. printf("I am ~Derived()\n");
32. }
33. public:
34. void SayHello();
35. void SayWorld();
36. };
38. void Derived::SayHello()
39. {
40. printf("Hello Derived\n");
41. }
42. void Derived::SayWorld()
43. {
44. printf("World Derived\n");
45. }
47. **int** main(void)
48. {
49. Base \*b1 = new Base;
50. Base \*b2 = new Derived;
51. Derived \*d = new Derived;
53. b1->SayHello();
54. b1->SayWorld();
56. b2->SayHello();
57. b2->SayWorld();
59. d->SayHello();
60. d->SayWorld();
62. delete d;
63. delete b2;
64. delete b1;
66. d= NULL;
67. b2 = NULL;
68. b1 = NULL;
70. return 0;
71. }

class Base

{

public:

Base()

{

printf("I am Base()\n");

}

virtual ~Base()

{

printf("I am ~Base()\n");

}

public:

virtual void SayHello()

{

printf("Hello Base\n");

}

void SayWorld()

{

printf("World Base\n");

}

};

class Derived : public Base

{

public:

Derived()

{

printf("I am Derived()\n");

}

virtual ~Derived()

{

printf("I am ~Derived()\n");

}

public:

void SayHello();

void SayWorld();

};

void Derived::SayHello()

{

printf("Hello Derived\n");

}

void Derived::SayWorld()

{

printf("World Derived\n");

}

int main(void)

{

Base \*b1 = new Base;

Base \*b2 = new Derived;

Derived \*d = new Derived;

b1->SayHello();

b1->SayWorld();

b2->SayHello();

b2->SayWorld();

d->SayHello();

d->SayWorld();

delete d;

delete b2;

delete b1;

d= NULL;

b2 = NULL;

b1 = NULL;

return 0;

}

输出结果：  
I am Base()  
I am Base()  
I am Derived()  
I am Base()  
I am Derived()  
Hello Base  
World Base  
Hello Derived  
World Base  
Hello Derived  
World Derived  
I am ~Derived()  
I am ~Base()  
I am ~Derived()  
I am ~Base()  
I am ~Base()  
  
  
11、阅读以下程序并给出执行结果

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/hackbuteer1/article/details/8476206)

1. class Bclass
2. {
3. public:
4. Bclass(**int** i , **int** j)
5. {
6. x = i;
7. y = j;
8. }
9. virtual **int** fun()
10. {
11. return 0;
12. }
13. protected:
14. **int** x , y;
15. };
17. class lclass : public Bclass
18. {
19. public:
20. lclass(**int** i , **int** j , **int** k) : Bclass(i , j)
21. {
22. z = k;
23. }
24. **int** fun()
25. {
26. return (x+y+z)/3;
27. }
28. private:
29. **int** z;
30. };
31. **int** main(void)
32. {
33. lclass obj(2,4,10);
34. Bclass p1 = obj;
35. cout<<p1.fun()<<endl;
37. Bclass &p2 = obj;
38. cout<<p2.fun()<<endl;
39. cout<<p2.Bclass::fun()<<endl;
41. Bclass \*p3 = &obj;
42. cout<<p3->fun()<<endl;
44. return 0;
45. }

class Bclass

{

public:

Bclass(int i , int j)

{

x = i;

y = j;

}

virtual int fun()

{

return 0;

}

protected:

int x , y;

};

class lclass : public Bclass

{

public:

lclass(int i , int j , int k) : Bclass(i , j)

{

z = k;

}

int fun()

{

return (x+y+z)/3;

}

private:

int z;

};

int main(void)

{

lclass obj(2,4,10);

Bclass p1 = obj;

cout<<p1.fun()<<endl;

Bclass &p2 = obj;

cout<<p2.fun()<<endl;

cout<<p2.Bclass::fun()<<endl;

Bclass \*p3 = &obj;

cout<<p3->fun()<<endl;

return 0;

}

输出结果：  
0  
5  
0  
5  
12、如何减少频繁分配内存（malloc或者new）造成的内存碎片？（10分）  
  
  
13、请写出strchr的实现（10分）  
函数功能：找出在字符串str中第一次出现字符ch的位置，找到就返回该字符位置的指针（也就是返回该字符在字符串中的地址的位置），找不到就返回空指针（就是NULL）  
const char\* strchr(const char\* str , char ch)

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/hackbuteer1/article/details/8476206)

1. const **char**\* strchr(const **char**\* str , **char** ch)
2. {
3. **char** \*p = NULL;
4. const **char**\* s = str;
5. for( ; \*s != '\0' ; ++s)
6. {
7. if(\*s == ch)
8. {
9. p = (**char** \*)s;
10. break;
11. }
12. }
13. return p;
14. }

const char\* strchr(const char\* str , char ch)

{

char \*p = NULL;

const char\* s = str;

for( ; \*s != '\0' ; ++s)

{

if(\*s == ch)

{

p = (char \*)s;

break;

}

}

return p;

}

14、请写出冒泡排序法算法（20分）  
void BubbleSort(int r[] , int n);

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/hackbuteer1/article/details/8476206)

1. void BubbleSort(**int** r[] , **int** n)
2. {
3. **int** i , j , temp;
4. for(i = 0 ; i < n - 1 ; ++i)
5. {
6. for(j = 0 ; j < n-i-1 ; ++j)
7. {
8. if(r[j] > r[j + 1])
9. {
10. temp = r[j];
11. r[j] = r[j + 1];
12. r[j + 1] = temp;
13. }
14. }
15. }
16. }