1. 若给出声明：

```
char c, *pc;
const char cc = 'a';
const char *pcc;
char *const cpc = &c;
const char * const cpcc = &cc;
char *const *pcpc;
```

则下面的赋值哪些是合法的？哪些是非法的？为什么？

```
(1) c = cc;              (10) *pc = "ABCD"[2];
(2) cc = c;              (11) cc = 'a';
(3) pcc = &c;            (12) *cpc = *pc;
(4) pcc = &cc;           (13) pc = *pcpc;
(5) pc = &c;             (14) **pcpc = *pc;
(6) pc = &cc;            (15) *pc = **pcpc;
(7) pc = pcc;            (16) *pcc = 'b';
(8) pc = cpcc;           (17) *pcpc = 'c';
(9) cpc = pc;            (18) *cpcc = 'd';
```

参考答案：

(1) 合法，c不是const类型的变量，可以赋值。

(2) 非法，cc是const类型的变量，不能赋值。

(3) 合法，pcc不是const类型的指针变量，可以指向非const类型的字符变量。

(4) 合法，pcc不是const类型的变量，赋值后指向的cc的类型为const char，同其要求指向的类型一致。

(5) 合法，pc 不是 const 类型的指针变量，赋值后指向的 c 的类型为 char，同其要求指向的类型一致。

(6) 非法，pc要求指向char类型的变量，不能用const char*类型的&c赋值。

(7) 非法，pc要求指向char类型的变量，不能用指向const char*类型的pcc赋值。

(8) 非法，pc要求指向char类型的变量，不能用指向const char*类型的cpcc赋值。

(9) 非法，cpc是const类型的变量，不能赋值。

(10) 合法，pc指向的是非const类型的变量，可以赋值，等价于*pc='C'。

(11) 非法，cc是const类型的变量，不能赋值。

(12) 合法，cpc指向的是非const类型的变量，可以赋值。

(13) 合法，pc是非const类型的指针变量，可以用char *类型的值*pcpc赋值。

(14) 合法，**pcpc代表的是非const类型的字符变量，可以任何字符类型的值赋值。

(15) 合法，*pc代表的是非const类型的字符变量，可以任何字符类型的值赋值。

(16) 非法，*pcc代表的字符是const类型的字符变量，不能赋值。

(17) 非法，*pcpc代表的是const类型的指针变量，不能赋值。

(18) 非法，*cpcc 代表的是 const 类型的只读变量，不能赋值。

2. C按优先级和结合性解释类型，下述声明是什么意思?

(1) typedef void VF_PC_RI(char*, int &)

(2) typedef VF_PC_RI* P_VF_PC_RI;

(3) typedef int &RIFFII(int, int);

(4) extern VF_PC_RI funca;

(5) extern P_VF_PC_RI ptra;

(6) extern void func1 (P_VF_PC_RI *);

(7) extern P_VF_PC_RI func2 (int c);

(8) P_VF_PC_RI func3 (P_VF_PC_RI a);

(9) typedef void (*(**VF_PA_P_PF_V(void))[ ]) (const int);

参考答案：

(1) 定义一个名为VF_PC_RI的类型，该类型定义了一个参数为(char*, int &)没有返回值的函数。

(2) 定义一个名为P_VF_PC_RI的类型，该类型定义了一个指向VF_PC_RI类型的指针。

(3) 定义一个名为RIFFII的类型，该类型定义了一个参数为(int ,int)返回一个引用的函数，该引用引用一个整型量。

(4) 说明一个类型为 VF_PC_RI 的函数 funca。

(5) 说明一个类型为P_VF_PC_RI的指针变量ptra。

(6) 说明一个没有返回值的函数func1，该函数的参数是一个指向P_VF_PC_RI类型的指针。

(7) 说明一个参数为int类型的函数func2，该函数的返回值是一个P_VF_PC_RI类型的指针。

(8) 说明一个参数为P_VF_PC_RI类型的函数func3，该函数的返回值是一个P_VF_PC_RI类型的指针。

(9) 定义一个名为 VF_PA_P_PF_V 的类型，该类型定义了一个没有参数的函数，该函数返回一个指针，该指针又指向另一个指针，被指向的指针指向一个数组，数组的每个元素存放一个函数指针，该函数指针指向一个参数为 const int 类型没有返回值的函数。

3. 下面g()函数的重载声明和定义是否会导致编译错误？

float g(int);

int g(int);

int g(int, int y=3);

int g(int, …);

int i = g(8);

参考答案：

(1) 不能定义返回类型仅和 float g(int)不同的函数 int g(int) 。

(2) g(8)在调用时出现二义性，无法确定是调用 int g(int, int y=3)还是 int g(int, …)。

4. 定义函数求 n（n>=1）个 double 类型的数的最大值 double max1(int n, …)。

参考答案：

```
double max1(int n, …) {
   double *p = (double *)(&n + 1);
   double v = p[0];
   for(int k = 1; k < n; k++) {
      if( v < p[k] ) v = p[k];
   }
   return v;
}
```

5. 编写函数 char *numConvert(int desBase, int srcBase, int num)，将 srcBase (2 <= srcBase <= 16) 进制数 num 转换为 desBase (2 <= desBase <= 16) 进制的字符串，返回字符串的首地址。

调用语句示例： char *p = numConvert(2, 8, 365); //将 8 进制数 365 转换为 2 进制字符串
printf(“%s \n”, p);

参考答案：

```
char *numConvert(int desBase, int srcBase, int num)
{
    char buf[100];
    int k = 0;
    /**/
    //将 srcNum 进制数 num 转换为 10 进制数 => num
    while(num != 0) {
        buf[k++] = num % 10;
        num /= 10;
    }
    while(--k >= 0) {
        num = num * srcBase + buf[k];
    }
    /**/
    //将 10 进制数 num 转换为 desNum 进制字符串
    k = 0;
    while(num != 0) {
        buf[k++] = num % desBase;
        num /= desBase;
    }
    char *p = new char [k+1];
    p[k--] = 0;
    for(int i = 0; k >= 0; i++, k--) {
        p[i] = buf[k];
        if(p[i] < 10) p[i] += '0';
        else p[i] = p[i] - 10 + 'A';
    }
    return p;
}
```

6. 集合类的头文件 set.h 如下，请定义其中的函数成员。

```
class SET {
    int *set;              //set 用于存放集合元素
    int card;              //card 为能够存放的元素个数
    int used;              //used 为已经存放的元素个数
public:
    SET(int card);         //card 为能够存放的元素个数
    ~SET( );
    int size( );           //返回集合已经存放的元素个数
    int insert(int v);     //插入 v 成功时返回 1，否则返回 0
    int remove(int v);     //删除 v 成功时返回 1，否则返回 0
    int has(int v);        //元素 v 存在时返回 1，否则返回 0
};
```

参考答案：

```
SET::SET(int card) {
    if (set = new int[card]) this->card = card;
    used = 0;
}

SET::~SET( ) {
    if( set ) {
       delete set;
       set = 0;
       card = used = 0;
    }
}
int SET::size( ) { return used; }
int SET::insert(int v) {
    if(used<card) { set[used++] = v; return 1;}
    return 0;
}
int SET::remove(int v) {
    int x;
    if( used > 0 ) {
       for(x = 0; x < used; x++) {
          if(set[x] == v) {
             used--;
             for(; x < used; x++) set[x] = set[x+1];
             return 1;
          }
          return 0;
       }
    }
    return 0;
}
```

```
int SET::has(int v) {
  for(int x = 0; x < used; x++) if(set[x] == v) return 1;
  return 0;
}
```

7. 二叉树类的头文件 node.h 如下，请定义其中的函数成员。

```
class NODE {
    char      *data;
    NODE   *left, *right;
public:
    NODE(char *data);
    NODE(char *data, NODE *left, NODE *right);
    ~NODE( );
};
```

参考答案：

```
NODE::NODE(char *data) {
  if( this->data = new char[strlen(data)+1] ) {
      strcpy(this->data, data);
      left = right = 0;
  }
}

NODE::NODE(char *data, NODE *left, NODE *right) {
    if( this->data = new char[strlen(data)+1]) {
        strcpy(this->data, data);
        this->left = left;
        this->right = right;
    }
}

NODE::~NODE( ) {
    if(left) left->~NODE( );
    if(right) right->~NODE( );
    if(data) { delete data; data = 0; }
}
```

8. 线性表通常提供元素查找、插入和删除等功能。以下线性表是一个整型线性表，表元素存放在动态申请的内存中，请编程定义整型线性表的函数成员。

```
class INTLIST {
    int   *list;          //动态申请的内存的指针
    int   size;           //线性表能够存放的元素个数
    int   used;           //线性表已经存放的元素个数
public:
```

```
    INTLIST(int s);        //s 为线性表能够存放的元素个数
    int insert(int v);     //插入元素 v 成功时返回 1，否则返回 0
    int remove(int v);     //删除元素 v 成功时返回 1，否则返回 0
    int find(int v);       //查找元素 v 成功时返回 1，否则返回 0
    int get(int k);        //取表的第 k 个元素的值作为返回值
    ~INTLIST( );
};
```

参考答案：

```
INTLIST::INTLIST(int s) {
    if( list = new int[s] ) {
        size = s;
        used = 0;
    }
}

INTLIST::~INTLIST(void) {
    if( list ) { delete list; list=0; size = used = 0; }
}

int INTLIST::insert(int v) {
    if( used < size) {
       list[used++] = v;
       return 1;
    }
    return 0;
}

int INTLIST::remove(int v) {
    for(int i = 0; i < used; i++) {
       if( list[i] == v ) {
          used--;
          for (; i < used; i++) list[i] = list[i+1];
          return 1;
       }
    }
    return 0;
}

int INTLIST::find(int v) {
    for(int i = 0; i < used; i++) {
        if(list[i] == v) return 1;
    }
    return 0;
}
```

1. 在 win32 x86 模式下，int *p; int **pp; double *q; 请说明 p、pp、q 个占几个字节的内存单元。

参考答案

不管是什么类型的指针，也不管是几重指针，它们存放的是一个物理地址。所以，p、pp、q 都是占 4 个字节的内存单元。

2. 常量 1、1.0、“1”的数据类型是什么？

参考答案

1	int
1.0	double
“1”	const char *

3. 语句：short int a[10]; short int *p = a; sizeof(a)等于 sizeof(p)吗？为什么？

参考答案

a 表示包含 10 个 short int 的缓冲区，p 变量存贮的是 a 的地址。

sizeof(a) = 20

sizeof(p) = 4 (win32 x86)

4. 类的构造函数和析构函数可以重载吗？为什么？

参考答案

构造函数：可以重载，因为构造函数的参数可以任意定义（除了 this）。

析构函数：不能重载，因为析构函数只能由 1 个 this 参数。

5. 写出下面 main()函数中每条指令的执行结果。

```
struct A {
        int    i;
        A(int v) { i = v; printf("A(%d) ",i); }
        A(const A &a) { i = a.i; printf("A(A&) "); }
        ~A( ) { printf("~A(%d) ",i); }
        operator int() const { printf("int() "); return i; }
        A &operator=(const A &a) {
            printf("=() ");
            i = a.i;    return *this;
        }
};
```

```
int main(void)
{
    A x = 1;
    x = 1;
    A y = x;
    y = x;
    x = 1 + x;
    A z(x+y);
    printf("%d %d", y, (int)y);
}
```

参考答案

```
int main(void)
{
    A x = 1;                    //A(1)
    x = 1;                      //A(1), =(), ~A(1)
    A y = x;                    //A(A&)
    y = x;                      //=()
    x = 1 + x;                  //int(), A(2), =(), ~A(2)
    A z(x+y);                   //int(), int(), A(3)
    printf("%d   %d",y,(int)y); //1(y.i), 1(int())
}                               //~A(3), ~A(1), ~A(2)
```

6. 字符串类的类型声明如下：

```
#include <string.h>
#include <iostream.h>
    class STRING {
    char *str;
public:
    int strlen( ) const;
    int strcmp(const STRING &s) const;
    STRING &strcpy(const STRING &s);
    STRING &strcat(const STRING &s);
    STRING(char *s);
    ~STRING( );
};
void main(void)
{
    STRING s1("I like apple");
    STRING s2(" and pear");
    STRING s3(" and orange");
    cout << "Length of s1=" << s1.strlen( ) << "\n";
    s1.strcat(s2).strcat(s3);
    cout << "Length of s1=" << s1.strlen( ) << "\n";
    s3.strcpy(s2).strcpy(s1);
```

```
    cout << "Length of s3=" << s3.strlen( ) << "\n";
}
```

试定义字符串复制及连接等函数成员，这些函数成员调用 C 的字符串运算函数。

参考答案

```
int STRING::strlen( )const{ return ::strlen(str); }
int STRING::strcmp(const STRING &s)const{return ::strcmp(str, s.str); }
STRING &STRING::strcat(const STRING &s) {
    int len = ::strlen(str) + ::strlen(s.str) + 1;
    char *t = str;
    if( str = new char[len] ) {
       ::strcat(::strcpy(str, t), s.str);
     }
     delete t;
    return *this;
}
STRING &STRING::strcpy(const STRING &s) {
    int len = ::strlen(s.str) + 1;
    delete str;
    if( str = new char[len] ) ::strcpy(str, s.str);
    return *this;
}
STRING::STRING(char *s) { if(str = new char[::strlen(s)+1]) ::strcpy(str, s); }
STRING::~STRING( ) { if (str) { delete str; str=0; } }
```

7. 完成下面类的成员函数。

```
class SEQUENCE;
class TREE {
    int    item;   //节点的值
    TREE *left, *right;
    friend SEQUENCE;
public:
    int getNodeNum( );          //返回节点总数
    int getNodes(int items[ ]);   //将所有的节点保存到items[ ]中
};
class SEQUENCE {
    int    *items;     //用于保存1个TREE中的所有节点
    int    size;       //items中元素的个数
 public:
    SEQUENCE(TREE &t);   //将t中的所有节点保存到items所指的缓冲区
};
```

参考答案

```
int TREE::getNodeNum( ) {
    int l = 0, r = 0;
    if(left) l = left->getnodes( );
```

```
        if(right) r = right->getnodes( );
        return l + r + 1;
    }
    int TREE::getNodes(int items[ ]) {
        int n = 0;
        if(left) n = left->getnodes(items);
        items[n++] = this->item;
        if(right) n += right->getnodes(items);
        return n;
    }
    SEQUENCE::SEQUENCE(TREE &t) {
        int m;
        size = t.getNodeNum( )
        items = new int[size];
        t.getNodes(items);
    }
```

8. 完成下面字典类的成员函数。

```
class DICT {
    char **const words;             //存放单词
    const int max;                  //字典可以存放单词的个数
    int pos;                        //当前可以存放单词的空闲位置
public:
    DICT(int max);                  //max 为最大单词个数
    DICT(const DICT &d);            //深拷贝构造
    DICT(DICT &&d) noexcept; //移动构造
    virtual ~DICT( ) noexcept;    //析构
    virtual DICT &operator=(const DICT &d);                 //深拷贝赋值
    virtual DICT &operator=(const DICT &&d) noexcept;   //移动赋值
    virtual int operator( )(const char *word) const;        //查找单词位置,-1 表示没找到
    virtual DICT &operator<<(const char *word);             //若字典中没有该单词则加入
    virtual DICT &operator>>(const char *word);             //删除字典中的这个单词,后面的单词往前移动
                                                            //字典中的单词保持连续存放
    virtual const char *operator[](int n) const;            //取出第 n(n>=0)个单词
};
```

参考答案

```
//max 为最大单词个数
DICT::DICT(int max): words(new char *[max]),max(max)
{
    pos = 0;
    for(int k = 0; k < max; k++)
    {
        words[k] = 0;
    }
}

//深拷贝构造
DICT::DICT(const DICT &d): words(0),max(0)
```

```
{
    *this = d;
}

//移动构造
DICT::DICT(DICT &&d) noexcept: words(0),max(0)
{
    *this = (DICT &&)d;
}

DICT::~DICT( ) noexcept
{
    if( words )
    {
        for(int k = 0; k < max; k++)
        {
            if( words[k] )
            { delete [] words[k];
              words[k] = 0;
            }
        }
        delete [] words;
        *(char ***)&words = 0;
        *(int *)&max = 0;
    }
}

//深拷贝赋值
DICT &DICT::operator=(const DICT &d)
{
    this->~DICT();
    *(char ***)&words = new char *[d.max];
    *(int *)&max = d.max;
    pos = d.pos;
    for(int k = 0; k < pos; k++)
    {
        words[k] = new char [strlen(d.words[k])+1];
        strcpy(words[k],d.words[k]);
    }
    for(int k = pos; k < max; k++) words[k] = 0;
    return *this;
}

//移动赋值
DICT &DICT::operator=(const DICT &&d) noexcept
{
    this->~DICT();
    *(char ***)&words = d.words;
    *(int *)&max = d.max;
    pos = d.pos;
    *(char ***)&d.words = 0;
    *(int *)&d.max = 0;
```

```
    return *this;
}

//查找单词位置,-1 表示没找到
int DICT::operator( )(const char *word) const
{
    int k = 0;
    while(k<pos && strcmp(word,words[k])!=0) k++;
    return k==pos? -1 : k;
}

//若字典中没有该单词则加入
DICT &DICT::operator<<(const char *word)
{
    if(pos<max-1 && (*this)(word) == -1)
    {
        words[pos] = new char [strlen(word)+1];
        strcpy(words[pos],word);
        pos++;
    }
    return *this;
}

//删除字典中的这个单词,后面的单词往前移动(字典中的单词保持连续存放)
DICT &DICT::operator>>(const char *word)
{
    int k = (*this)(word);
    if( k >= 0 )
    {
        delete [] words[k];
        while(k < pos - 1)
        {
            words[k] = words[k+1];
            k++;
        }
        words[--pos] = 0;
    }
    return *this;
}

//取出第 n(n>=0)个单词
const char *DICT::operator[](int n) const
{
    return n>=0 && n<pos? words[n] : 0;
}
```

9. 分析 mian( )函数中每条语句的变量 i 的值。

```
int x = 1;
int y = ::x + 1;
```

```
struct A {
    int x;
    static int y;
    A &operator+=(A &a) { x += a.x; y += a.y; return *this; }
    operator int() { return x + y; }
    A(int x = ::x+1, int y = ::y + 10): x(x) { this->y = y; }
};
int A::y = 100;

int main() {
    A a(1,2), b(3), c;
    int i, *p = &A::y;
    i = b.x + b.y;
    i = *p;
    i = c;
    i = a + c;
    i = b += c;
    i = ((a+=c)=b)+10;
}
```

参考答案

	i	**::x**	**::y**	**A::y**	**a.x**	**b.x**	**c.x**
a(1, 2)		1	2	2	1		
b(3)				12		3	
c				12			2
i = b.x + b.y	15						
i = *p	12						
i = c	14						
i = a + c	27						
i = b += c	29			24		5	
i = ((a += c) = b) + 10	63			48 48	3 5		

1. 虚函数、纯虚函数可以定义为 static 成员函数吗？为什么？

参考答案

不能。因为虚函数、纯虚函数只能用来说明类的实例成员函数（含有 this 指针，用于实现多态），而 static 成员函数不属于任何对象，不含 this 指针。

2. 构造函数、析构函数可以定义为虚函数和纯虚函数吗？为什么？

参考答案

构造函数的作用是构造一个确定的对象，没有多态性，而虚函数和纯虚函数用于实现动态多态，所以构造函数不能声明为虚函数和纯虚函数。
析构函数可以定义为虚函数，这样可以实现析构时的动态多态。析构函数可以定义为纯虚函数，这时派生类不可能来实现基类的析构函数，因此无法实例化。

3. 分析下面的程序，指出错误之处（解释错误原因），并写出 main()函数中每条正确的指令的屏幕输出结果。

```
struct A {
    int a = 0;
    int x = 1;
    void f() { cout << "A::f()"; }
    virtual void g() { cout << "A::g()"; }
    A(int x) { }
};
struct B: A {
    int x = 11;
    int y = 12;
    virtual void f() { cout << "B::f()"; }
    void g() { cout << "B::g()"; }
    void h() { cout << "B::h()"; }
    B(int x): A(x) { }
};
struct C: B {
    int x = 21;
    int y = 22;
    int z = 23;
    void f() { cout << "C::f()"; }
    void g() { cout << "C::g()"; }
    virtual void h() { cout << "C::h()"; }
    C(int x): B(x) { }
} c(1);
int main() {
```

```
    A *p = &c;
    p->f();
    p->g();
    p->h();
    cout << p->a;
    cout << p->x;
    cout << p->y;
    cout << p->z;
    /**/
    B *q = &c;
    q->f();
    q->g();
    q->h();
    cout << p->a;
    cout << q->x;
    cout << q->y;
    cout << q->z;
}
```

参考答案

只有 main()函数中存在错误的指令。

```
int main() {
    A *p = &c;
    p->f();             //A::f()
    p->g();             //C::g()
    p->h();             //错误, 类 A 中没有 h()
    cout << p->a;       //0 (A::a)
    cout << p->x;       //1 (A::x)
    cout << p->y;       //错误, 类 A 中没有数据成员 y
    cout << p->z;       //错误, 类 A 中没有数据成员 z
    /**/
    B *q = &c;
    q->f();             //C::f()
    q->g();             //C::g()
    q->h();             //B::h()
    cout << p->a;       //0 (A::a)
    cout << q->x;       //11 (B::x)
    cout << q->y;       //12 (B::y)
    cout << q->z;       //错误, 类 B 中没有数据成员 z
}
```

4. 指出如下各类可访问的成员及成员的访问权限。

```
class A {
    int  a;
```

```
    protected:
        int  b;
    public:
        int  c;
        ~A( );
    };
    class B: A {
        int  a;
    protected:
        int b;
        A::b;
    public:
        int  c, d;
    };
    class C: protected A {
        int  a;
    protected:
        int  b, e;
    public:
        int  g;
        A::c;
    };
    struct D: B, C {
        int a;
    protected:
        int b, f;
    public:
        int e, g;
};
```

参考答案

class A

private: a
protected: b
public: c, ~A()

class B

private: a, A::(c, ~A())
protected: b, A::b
public: c, d

class C

private: a
protected: b, e, A::(b, ~A())
public: g, A::c

class D

private:

protected: b, f, B::(~~b,~~A::b), C::(b, e, A::(b, ~A()))
public: a, e, g, B::(c, d), C::(g, A::c)

5. 指出如下程序的错误之处及其原因：

```
class A {
      int x;
      virtual int f( ) { return 0; }
      virtual int g( ) = 0;
protected:
      int  y;
public:
     virtual A( ) { }
} a;
struct B: A {
      A::x;
      using A::y;
      long f( )  { return 1L; };
      int g(int) { return 1; }
} b;
A *p = new A;
B *q = new B;
int f(A, B);
A g(B &);
int h(B *);
```

参考答案

(1) virtual A( )中去掉virtual，构造函数不能为虚函数。
(2) 不能定义对象a，因为类A是抽象类。
(3) B中，A::x 错，因为 A::x不能访问。
(4) B中，using A::y 错，因为不能将 A::x原来的 protected 属性扩大为 public。
(5) B中的long f()中的long应改为int，在基类和派生类中不能定义原型相同、只有返回值不同的成员函数。
(6) 不能定义对象b，因为类B是抽象类（含有纯虚函数g( )）。
(7) 不能 new A，因为类A是抽象类。
(8) 不能 new B，因为类B是抽象类。
(9) 不能为函数int f(A, B)产生实参对象，因为类A、B是抽象类。
(10) 函数A g(B &)不能返回A类对象，因为类A是抽象类。

6. 指出如下程序中main()中每行语句的输出结果。

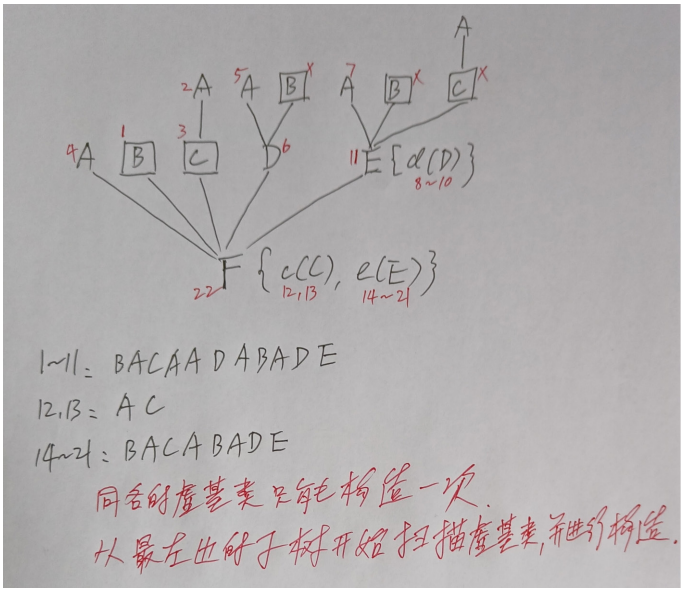
```
struct A { A( ) { cout << 'A'; } };
struct B { B( ) { cout << 'B'; } };
```

```
struct C: A { C( ) { cout << 'C'; } };
struct D: A, virtual B { D( ) { cout << 'D'; } };
struct E: A, virtual B, virtual C {
        D d;
        E( ) { cout << 'E'; }
};
struct F: A, virtual B, virtual C, D, E {
        C  c;
        E  e;
        F( ) { cout << 'F'; }
};
void main(void)
{
     A  a;
     B  b;
     C  c;
     D  d;
     E  e;
     F  f;
}
```

参考答案

```
A  a;       //A
B  b;       //B
C  c;       //AC
D  d;       //BAD
E  e;       //BACABADE
F  f;       //BACAADABADEACBACABADEF
```

F 的构建过程见下图：



1. mutable 成员一般在什么情况下定义？mutable 成员可以同时定义为 static、const、volatile 或它们的组合吗？说明理由。

参考答案

mutable 用于修饰类的非静态数据成员。它使得 const 对象的 mutable 数据成员可以被修改。mutable 的一个应用是：当实例成员函数的参数表后出现 const 时，该函数不能修改 this 对象的非静态数据成员，但如果数据成员是 mutable 的，则该数据成员就可以被修改。mutable 不能与 const、static 连用。

2. 类的实例成员指针和静态成员指针有什么不同？

参考答案

实例成员指针：是一个偏移量，不能移动和参与运算，需要结合对象或对象指针来使用。
静态成员指针：实际上是普通的指针，存放的是成员的物理地址，不需要结合对象使用。

3. 分析如下定义是否正确，并指出错误原因。

```
struct A {
    char *a, b, *geta( );
    char A::*p;
    char *A::*q( );
    char *(A::*r)( );
};
int main(void) {
    A a;
    a.p = &A::a;
    a.p = &A::b;
    a.q = &A::geta;
    a.r = a.geta;
    a.r = &a.geta;
    a.r = &A::geta;
}
```

参考答案

```
void main(void) {
    A a;
    a.p = &A::a;      //错：不能将&A::a的类型char *A::*转换为a.p的类型char A::*
    a.p = &A::b;      //对
    a.q = &A::geta;   //错：a.q是1个函数，不是函数指针
    a.r = a.geta;     //错：a.r是函数成员指针，a.geta既不是取函数geta的地址，也不是调用geta函数
    a.r = &a.geta;    //错：a.r是函数成员指针，&a.geta不是函数成员指针
    a.r = &A::geta;   //对
}
```

4. 分析如下定义是否正确，并指出错误原因。

```
struct A {
    static int x = 1;
    static const int y = 2;
    static const volatile int z = 3;
    static volatile int w = 4;
    static const float u = 1.0f;
};
static int A::x = 11;
int A::y = 22;
int volatile A::z = 33;
int volatile A::w = 44;
const float A::u = 55.0f;
```

参考答案

```
struct A {
    static int x = 1;                   //错：static，只有const 整型变量（不能带volatile）才能设置缺省值
    static const char y = 'a';          //对
    static const volatile int z = 3;    //错：static，只有const 整型变量（不能带volatile）才能设置缺省值
    static volatile int w = 4;          //错：static，只有const 整型变量（不能带volatile）才能设置缺省值
    static const float u = 1.0f;        //错：static，只有const 整型变量（不能带volatile）才能设置缺省值
};
static int A::x = 11;                   //错：不能写static（只能 int A::x = 11）
char A::y = 22;                         //错：y已经有缺省值（表示已经初始化）
int volatile A::z = 33;                 //错：应当为 int const volatile A::z = 33
int volatile A::w = 44;                 //对
const float A::u = 55.0f;               //对
```

5. 分析如下定义是否正确，并指出错误原因。

```
class A {
    static int *j, A::*a, i[5];
public:
    int x;
    static int &k, *n;
};
int y = 0;
int A::i[5] = {1, 2, 3};
int *A::j = &y;
int A::*j = &A::x;
int A::*A::a = &A::x;
int &A::k = y;
int *A::n = &y;
```

参考答案

全部都是对的，没有错误。

6. 分析如下定义是否正确，并指出错误原因。

```
class A {
    int a;
    static friend int f( );
    friend int g( );
public:
    friend int A( );
    A(int x): a(x) { };
} a(5);
int f( ) { return a.a; }
int g( ) { return a.a; }
```

参考答案

语句 friend int A( ) 错误：构造函数不能有返回值，其次friend只能修饰不属于类的函数。

7. 完成下面堆栈类 STACK 和 REVERSE 类的函数成员定义。

```
class STACK {
    const int max;          //栈能存放的最大元素个数
    int   top;              //栈顶元素位置
    char *stk;
public:
    STACK(int max);
    ~STACK( );
    int push(char v);       //将v压栈，成功时返回1，否则返回0
    int pop(char &v);       //弹出栈顶元素，成功时返回1，否则返回0
};
class REVERSE: STACK {
public:
    REVERSE(char *str); //将字符串的每个字符压栈
    ~REVERSE( );          //按逆序打印字符串
};
void main(void) {
    REVERSE a("abcdefg");
}
```

参考答案

```
STACK::STACK(int max): top(0), max( (stk=new char[max])? max : 0) { }
STACK::~STACK( ) { if(stk) { delete stk; stk = 0; *(int *)&max =0; } }
int STACK::push(char v) {
    if (top >= max) return 0;
    stk[top++] = v;
    return 1;
}
```

```
int STACK::pop(char &v) {
    if (top <= 1) return 0;
    v = stk[--top] = v;
    return 1;
}

REVERSE ::REVERSE(char *str): STACK(strlen(str)) {
    for(int s = 0; s < strlen(str); s++) {
      push(str[s]);
    }
}
REVERSE ::~REVERSE( ) {
    char c;
    while( pop(c) ) printf("%c", c);
}
```

8. 找出下面的错误语句，说明错误原因。然后，删除错误的语句，指出类 A、B、C 可访问的成员及其访问权限。

```
class A {
    int a1;
protected:
    int a2;
public:
    int a3;
    ~A( ) { };
};
class B: protected A {
    int b1;
protected:
    int b2;
public:
    A::a1;
    A::a2;
    int b3;
    ~B( ) { };
};
struct C: private B {
    int c1;
protected:
    int c2;
    B::A::a2;
    A::a3;
public:
    using B::b2;
    int c3;
    int a3;
    ~C( ) { };
};
```

```
int main() {
    C c;
    cout << c.b2;
    cout << c.B::b2;
}
```

参考答案

错误的语句：

```
class B:   A::a1;              //error, 无法访问 A::a1
class C:   int a3;             //error, 与“A::a3;”冲突
main():    cout << c.B::b2;    //B 是 private, 不能访问
```

类成员访问权限：

class A

private: a1
protected: a2
public: a3, ~A()

class B

private: b1
protected: b2; A::(a3, ~A())
public: b3, ~B(); A::(a2)

class C

private:
protected: B::A::(a2, a3)
public: c1, c2, c3, ~C(); B::(b2)

1. 如果有 3 个异常处理 hi 过程：catch(…)、catch(const void *)、catch(int *)，应如何摆放它们的位置？catch(int *)可以放在 catch(const void *)后面而不影响其捕获异常吗 ？

答案： (1) 根据下面的顺序摆放：catch(int *)、catch(const void *)、catch(…)。
(2) 将 catch(int *)放在 catch(const void *)后面，将导致 catch(int *)失效。

2. int x，显式转换(int)x 的结果是左值还是右值？

答案： 右值。

3. 对于全局变量 const int x = 0，能够使用 *const_cast<int *>(&x) = 3 修改 x 的值吗？用 *(int *)(&x) = 3 可以修改 x 的值吗？

答案： 都不能。非类内定义的简单类型变量即使转换为左值，也不能修改其值（受到页面保护机制的保护）。

4. 说明 static_cast 和 const_cast 的区别。

答案： static_cast 和 const_cast 同 C 语言的强制类型转换用法基本相同，但 static_cast 不能修改源类型中的 const 和 volitale 属性，const_cast 则可以。

5. 对于 Lambda 表达式：auto f = [x] (int y) -> int { return ++x + y; }，解释 [x] (int y) 中的 x 和 y 的意义，并说明为什么 f 名义上是一个函数，但实际上是一个对象。

答案： [x]表示捕捉 lambda 函数体外变量 x 的值，并根据这个值创建匿名类的实例成员变量 x。(int y)表示匿名类的( )运算符的重载函数 operator( )(…)的调用参数，即 int operator( )(int y)。

6. 类模板的实例化有哪几种方式？

答案： 2 种方式：显式实例化和隐式实例化。
例如，对于类模板 MAT<T>，
显式实例化：template class A<1>;（等价 template class A<int>;）
隐式实例化：MAT<int> a(2,3);（实例化类模板 MAT 并且创建对象 a）

7. 分析下面的程序，指出变量 g1～g4、a1～a6 中，哪些变量的地址是相同的（指向同一个对象）？执行完指令“float &a5 = f<float>();”和“float a6 = f<float>();”后，各变量的值是多少？

```
template<typename T, int x=0>
T g = T(10 + x);
template float g<float>;
template<typename T>
```

```
T &f( ) {
    T &a = g<T, 0>;
    return ++a;
}
float g1 = g<float>;
float &g2 = g<float>;
const float &g3 = g<float>;
const float &g4 = g<float, 4>;
int main()
{
    float a1 = g<float>;
    float &a2 = g<float>;
    const float &a3 = g<float>;
    const float &a4 = g<float, sizeof(float)>;
    float &a5 = f<float>();
    float a6 = f<float>();
}
```

答案： g2、g3、a2、a3、a5 指向同一个对象（显式创建的、初始值为 10.0f 的匿名变量）。

g4、a4 指向同一个对象（隐式创建的、初始值为 14.0f 的匿名变量）。

g1、a1、a6 分别指向不同的存贮单元。

执行完“float &a5 = f<float>()”：a1=g1=10，g4=a4=14，g2=g3=a2=a3=a5=11

执行完“float a6 = f<float>()”： a1=g1=10，g4=a4=14，g2=g3=a2=a3=a5=a6=12

分析：

“template float g<float>”显式地创建全局 float 类型的匿名变量（值为 10.0f）。

“float g1 = g<float>;”创建变量 g1，将前面创建的值为 10.0f 的匿名变量的值拷贝到 a1。

“float &g2 = g<float>;”引用变量 g2 引用前面创建的值为 10.0f 的匿名变量。

“const float &g3 = g<float>;” 引用变量 g3 引用前面创建的值为 10.0f 的匿名变量。

“const float &g4 = g<float, 4>;”隐式创建全局 float 类型的匿名变量（值为 14.0f）。

“float a1 = g<float>;”创建变量 a1，将前面创建的值为 10.0f 的匿名变量的值拷贝到 a1。

“float &a2 = g<float>;”引用变量 a2 引用前面创建的值为 10.0f 的匿名变量。

“const float &a3 = g<float>;”引用变量 a3 引用前面创建的值为 10.0f 的匿名变量。

“const float &a4 = g<float, sizeof(float)>;”引用变量 a4 引用前面创建的值为 14.0f 的匿名变量。

“float &a5 = f<float>();”f()返回前面创建的值为 10.0f 的匿名变量的引用，并将该匿名变量的值加 1，引用变量 a5 引用该匿名变量。

“float a6 = f<float>();”f()返回前面创建的初始值为 10.0f 的匿名变量的引用，并将该匿名变量的值加 1，并将该值拷贝到新的变量 a6。

8. 设计一维数组模板 ARRAY，可以实例化各种简单类型的一维数组。在模板的成员函数中需要考虑抛出异常（如内存不足、数组下标越界等），并在 mian()中进行简单的测试（包括捕获异常处理）。

答案： 参考实验四。

9. 设计二维数组模板 ARRAY2，可以实例化各种简单类型的二维数组。模板中包含一个非类型形参，表示构造函数的一个参数的缺省值，用于初始化数组元素的值。并给出任意一条实例化模板的语句。

答案： 参考实验四。

计算机科学与技术学院 2016-2017 学年第 1 学期　考试试卷

面向对象程序设计　　A 卷　开卷

姓名_____班级_____学号_____考试日期_____

题号	一	二	三	四	五	六	总分	核对人
题分	15	20	20	15	15	15	100	马光志
得分								

得分	评卷人

一、选择题(每小题 3 分，共 15 分)
答题卡见右，以答题卡为准。

1	2	3	4	5
C	B	B	A	D

1. 对于如下程序：

```
#include <stdio.h>
struct A{
    virtual int f( ) { printf("A"); }
    virtual int g( ) { printf("B"); }
} a, *p;
struct B: A {
    int f( ) { A::f( ); printf("C"); }
    int g( ) { printf("D"); }
};
void main( ) { p=&b; p->f( ); p->g( ); }
```

则程序的输出为_____：

A. AB　　**B. CD**

C. ACD　　**D. ABCD**

2. 如下程序：

```
#include <stdlib.h>
struct A { A( ) { } }a;
void main( ) { A b; exit(0 ); }
```

则关于对象 a, b 的析构，如下哪个叙述正确_____：

A. 无析构函数都没有析构;　　**B.** a 析构了但是 b 没析构;

C. b 析构了但是 a 没析构;　　**D.** 都析构了;

3. 对于定义“const char *&g( );”，如下哪个语句是错误的_____:

A. g( )= "abcde";　　**B.** *g(　)= 'A';

C. const char *p=g(　);　　**D.** const char *&q =g(　);

4. 对于如下定义：

```
struct A { virtual int f( ) { return 1;} }a;
struct B: A {
    int f( ) const { return 2; }
    int f( ) volatile { return 3; }
    int f( ) const volatile { return 4; }
} c;
int main(int argc, char* argv[ ]) { A *p = &c;   return p->f( ); }
```

主函数 main 的返回值是_____:

A. 1　　B. 2

C. 3　　D. 4

5. 关于运算符重载的叙述哪个正确_____:

A. 可以改变优先级和结合性　　B. 可以改变优先级，不能改变结合性

C. 不能改变优先级，可改变结合性 D. 都不能改变

得分	评卷人

二、在作用域运算最多允许一层（即不允许两层运算如 A::B::c 及两层以上运算）的情况下，指出各类可访问的成员及其访问权限(20)。

```
class A{                    |A 的成员
    int   a;                |private:
protected:                  |     int a;
    int   b, e;             |protected:
public:                     |     int b, e;
    int   c, d;             |public:
};                          |     int c, d;

class B: protected A{       |B 的成员
    int   d;                |private:
protected:                  |     int d;
    int   c, e;             |protected:
public:                     |     int c, e, A::(b, d, e)
    A::c;                   |public:
    int   f, b;             |     int f, b, A::c
};

class C: public A {         |C 的成员
    int   g;                |private:
protected:                  |     int g;
```

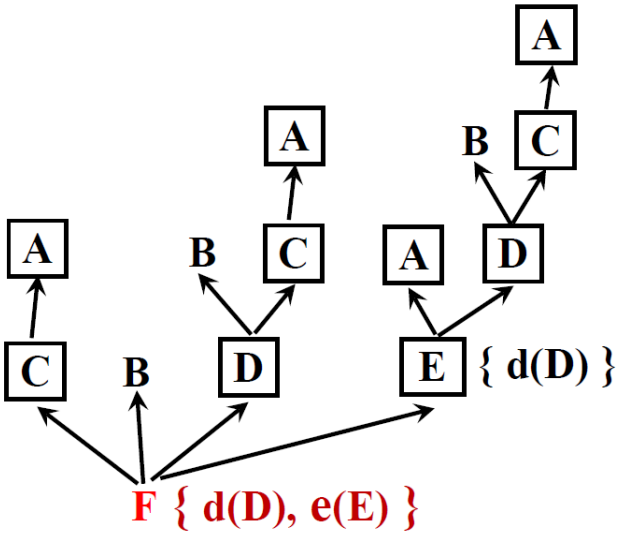
```
    int   h, d;              |protected:
public:                      |     int h, d, A::(b, e)
    int   i;                 |public:
};                           |     int i, A::(c, d)
struct D: protected B, C{    |D 的成员
    int   j;                 |private:
protected:                   |protected:
    int   k;                 | int k,B::(c,e,f,b,A::(b,c,d,c)), C:(h,d, A::(b,c))
public:                      |public:
    int   n, d;              |     int j, n, d, C::(i, A:(c, d))
};
```

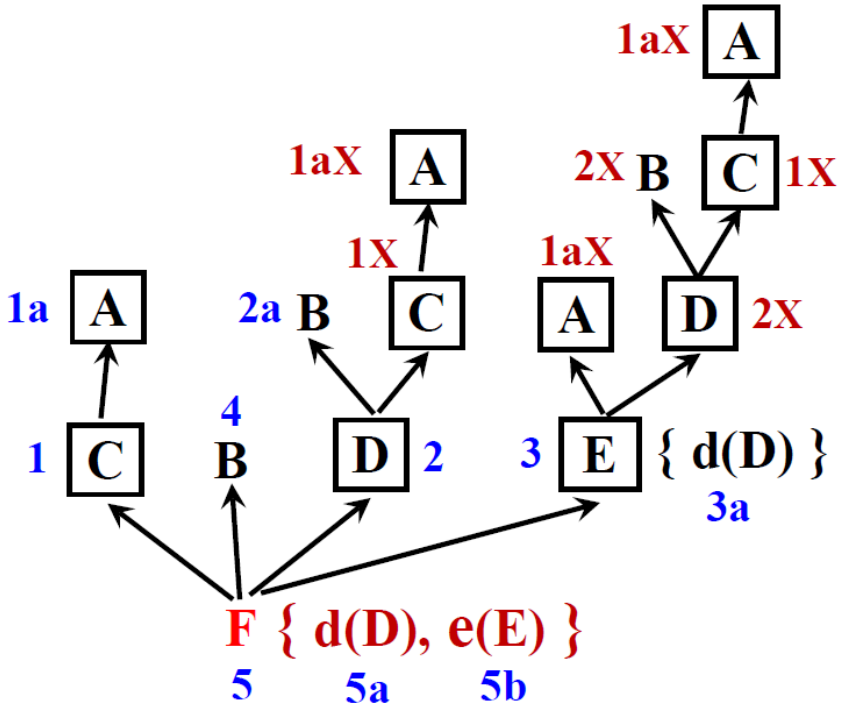
得分	评卷人

三、指出 main 中每行的输出结果 (前四题每题 3 分，后两题每题 4 分，共 20 分)

```
#include <iostream.h>
struct A {A( ) { cout<<'A'; } };
struct B {B( ) { cout<<'B'; } };
struct C: virtual A { C( ) { cout<<'C'; } };
struct D: B, virtual C { D( ) { cout<<'D'; } };
struct E: virtual A, virtual D {
    D   d;
    E( ): A( ) { cout<<'E'; }
};
struct F: virtual C, B, virtual D, virtual E {
    D d;   E e;
    F( ) { cout<<'F'; }
};
void main( ) {
    A   a;   cout<<'\n';   //输出=A
    B   b;   cout<<'\n';   //输出=B
    C   c;   cout<<'\n';   //输出=AC
    D   d;   cout<<'\n';   //输出=ACBD
    E   e;   cout<<'\n';   //输出=ACBDACBDE
    F   f;   cout<<'\n';   //输出=ACBDACBDEBACBDACBDACBDEF
}
```

下面给出 **F f** 的分析：





F 输出：ACBDACBDEBACBDACBDACBDEF

E 输出：ACBD ACBD E

同名的虚基类只能构造 1 次；

从最左边的子树开始扫描虚基类，并进行构造。

得分	评卷人

四、指出以下程序的语法错误及其原因 (每错约 1 分，共 15 分)

```
class A {
    int &a;
protected:
    const int &b;
    ~A( ) { }
public:
    int c;
    virtual A&   (*g)( );  //(1)virtual 不能用于定义数据成员
    A(int x) { b = x; };   //(2)没在函数体前初始化 a 和 b, (3)不能在体内初始 b
} a = (1, 2, 3);           //(4)a 不能调用私有的析构函数~A( )

class B: A {
    int d;
public:
    A::b;                  //(5)不能改变访问权限，只能恢复（有问题）
    static int operator( )(int) { return 3; }; //(6)运算符( )只能为实例函数成员
    B(int x, int y, int z):A(x) { d=x+y+z; };
} b(2, 3, 7);

struct C: B{               //(7)编译程序无法自动生成构造函数 C( )
    int z;
protected:
    virtual ~C( ) {   };
} c;                       //(8)c 不能调用保护的析构函数~C( )

void   main( ) {
    int *A::*p = &c.z;     //(9)不能将 int *类型的值赋给 int *A::*类型的变量 p
    int   i = a.b;         //(10)main 不能访问保护成员 A::b（有问题）
    i = a;                 //(11)对象 a 无法转换为整数
    i = b.b;               //(12) main 不能访问私有成员 B::b
    i = c.d;               //(13) main 不能访问私有成员 C::d
    i = b.*p;              //(14)不能将 int *类型的值赋值给整型变量 i
    return 1;              //(15)main 无需返回值
}
```

得分	评卷人

五、请填入学号最后一位十进制数字，指出 main 函数中变量 i 在每条赋值语句执行后的值 (每小题 2.5 分，共 15 分)

```
int    x = 学号最后一位十进制数      ,  y=x+3;
struct A {
    int x;
    static int &y;
public:
    operator int( )const { return x + y; }
    int &v(int &x) {
        for(int y=1; x<201; x^=y, y++)
            if(x>200) { x -= 31; y -= 2;}
         return ++x;
    }
    A &operator++( ) { ++x; ++y; return *this; }
    A(int x = ::x + 2, int y = ::y + 3) { A::x = x;   A::y = y; }
};

int & A::y = ::x;

void main( ) {
    A   a(2, 3),   b(a),   c;
    int   i,   &j=i,   A::*p=&A::x;
    i = a.y;                    //i=
    j = a.x++;                  //i=
    i = a.*p;                   //i=
    i = ++a;                    //i=
    i = b.y + ::y;              //i=
    (b.v(i) = 2) += 3;          //i=
}
```

答：

学号	i=a.y	j=a.x++	i=a.*p	i=++a	i=b.y+::y	(b.v(i)=2)+=3
0	6	2	3	11	10	5
1	7	2	3	12	12	5

2	8	2	3	13	14	5
3	9	2	3	14	16	5
4	10	2	3	15	18	5
5	11	2	3	16	20	5
6	12	2	3	17	22	5
7	13	2	3	18	24	5
8	14	2	3	19	26	5
9	15	2	3	20	28	5

分析：

	::y	::x, A::y	a.x	b.x	c.x	i
	n+3	n				
a(2, 3)		3	2			
b(a)		3		2		
c <=> A(5, n+6)		n+6			5	
i = a.y						n+6
j = a.x++			3			2
i = a.*p						3
i = ++a		n+7	4			n+11
i = b.y + ::y						2n+10
(b.v(i)=2) += 3						5

得分	评卷人

六、N 个顶点的无向图 MAP 最多有 N*(N-1)条边，设顶点的编号为 0, 1, …, N-1，每条边由其中任意两个顶点连接而成，试定义如下无向图类中的成员函数。(每小题 2.5 分，共 15 分)

```
class MAP {
    int (*const e)[2];    //边集指针 e,边 x 的顶点为 e[x][0]和 e[x][1]
    const int n;          //图的顶点个数
```

```
    int c;                  //图实际已有的边的个数
public:
    MAP(int n);    //图最多 n 个顶点，假设图初始时无边
    MAP(const MAP& m);                  //深拷贝构造函数
    MAP &operator=(const MAP& m);   //深拷贝赋值函数
    MAP &operator( )(int v0, int v1);   //连接顶点 v0 和 v1 成边,设 v0<v1
    int (*operator[ ](int x))[2];       //取图中的边 x
    ~MAP( );                            //析构函数
};
```

答：

```
MAP::MAP(int n): e(new int[n*(n-1)][2]),n(e? n:0), c(0) { }

MAP::MAP(const MAP& m): e(new int[m.n*(m.n-1)][2]),n(e?m.n:0){
    for(c=0; c<m.c; c++){
        e[c][0] = m.e[c][0];
        e[c][1] = m.e[c][1];
    }
}

MAP& MAP::operator=(const MAP &m) {
    if(e) delete e;
    *(int (**)[2])&e = new int[m.n*(m.n-1)][2];
    *(int *)&n = e?m.n:0;
    for(c=0; c<m.c; c++){
        e[c][0] = m.e[c][0];
        e[c][1] = m.e[c][1];
    }
    return *this;
}

MAP& MAP::operator( )(int v0, int v1) {//连接顶点 v1 和 v2 成为边
    int t=v0;
    if(v0==v1 || v0<0 || v0>=n || v1<0 || v1>=n) return *this;
```

```
    if(v0>v1) { v0 = v1; v1 = t; }
    for(t=0; t<c; t++)    if(e[t][0]==v0 && e[t][1]==v1) return *this;
    e[c][0] = v0;
    e[c++][1] = v1;
    return *this;
}

MAP::~MAP( ) {    //析构函数
    if(e) { delete e; *(int (**)[2])&e=0; *(int *)&n=0; c=0; }
}

int (*MAP::operator[ ](int x)) [2] {   return e + x;   }
```

C++模拟试卷-2016

一、选择题（每小题 3 分，共 15 分）

1. 类 A 是一个包含纯虚函数的抽象类，下列说明语法正确的是__C____

A. A a;　　B. A f( );

C. A &f( );　　D. A f(A);

2. 使用 exit 退出程序，关于对象自动析构，哪个叙述正确__B____

A.不析构全局对象但析构局部对象

B.析构全局对象但不析构局部对象

C.全局对象和局部对象都不析构

D.全局对象和局部对象都析构

3. 对于如下程序：

```
#include <stdio.h>
struct A {
    A( ) { printf("别理我"); }
    A(const cha r *s) { printf(s); }
} a("烦着呢");
A f( ) { printf("一边去"); return a; }
void main(void) { A f( ); }
```

关于程序的输出，哪个叙述是正确的 ___A____

A. 输出为“烦着呢”；　　B. 输出为“烦着呢 别理我”；

C. 输出为“一边去”；D. 输出为“烦着呢 一边去”；

4. 对于 int x；如下运算错误的是___A___

A. x++ ++　　B.++ ++x

C. (++x)++　　D. ++ (++x)

5. 对于 int x，int &y 最好引用如下哪个表达式__A____

A. x+=3;　　B. x+4 ;

C. x++;　　D. (--x)--;

二、指出各类可访问的成员及其访问权限 （20）。

```
class A {
    int  a;
protected:
    int  b, f;
public:
    int  c, d;
};
```

A:
private:　a
protected:　b, f
public:　c, d

```
class B: protected A {
    int d;
protected:
    int c, e;
public:
    int f;
};
```

B:
private: d
protected: c, e, A::(b, c, d, f)
public: f

```
class C: public A {
    int g;
protected:
    int h, d;
public:
    int c, i;
};
```

C:
private: g
protected: h, d, A::(b, f)
public: c, i, A::(c, d)

```
struct D: B, public C {
    int j;
protected:
    int k, c;
public:
    int n;
};
```

D:
private:
protected: k, c, B::(c, e, A::(b, c, d, f)), C::(h, d, A::(b, f))
public: j, n, B::(f), C::(c, i, A::(c, d))

三、指出 main 中每行的输出结果 （前四题每题 3 分，后2题每题4分）

```
#include <iostream>
struct A { A( ) { cout << 'A'; } };
struct B { B( ) { cout << 'B'; } };
struct C: A { C( ) { cout << 'C'; } };
struct D: B, virtual C { D( ) { cout << 'D'; } };

//struct E: virtual A, virtual D {
struct E: A, virtual D {
    D d;
    E( ): A( ) { cout << 'E'; }
};
struct F: B, virtual C, E, virtual D {
    D d;
    F( ) { cout << 'F'; }
};

void main( ) {
```

```
    A a; cout << "\n";      //输出= A
    B b; cout << "\n";      //输出= B
    C c; cout << "\n";      //输出= AC
    D d; cout << "\n";      //输出= ACBD
    E e; cout << "\n";      //输出= ACBDAACBDE
    F f; cout << "\n";      //输出= ACBDBAACBDEACBDF
}
```

四、指出以下程序的语法错误及其原因（每错约 1分，共 15 分）

```
class A {
    int a;
protected:
    const int &b;
    ~A( ) { }
public:
    int c;
    virtual A (*g)(int);    //(1) g是一个变量，不能定义为virtual
    A(int x) { a = x; };    //(2) b没有初始化
} x = (4, 3);               //(3) x不能析构

class B: A {
    int d;
public:
    A::b;   //
    friend int operator( )(int) { return 2; };  //(4) 不能声明为友元
    B(int x, int y, int z) { d = x + y + z; };  //(5) 没有说明基类的构造方法
} b(5, 6, 7);   (6) 不能生成对象b

struct C: B {
    int z;
public:
    ~C(int x) { z = x; };
} c;   //(7) 没有构造函数；(8) 不能生成对象c
void main( ) {
    int A::*p = &c.z;       //(9) 成员指针不能指向物理地址
    int i = x.b;            //(10) 不能访问类A的保护成员
    i = x;                  //(11) 类A没有强制类型转换函数
    i = b.b;                //
    i = i + c.d;            //(12) 不能访问类B的私有成员d
    i = b.*p;               //(13) p不是类A的成员指针
}
```

五、请填入学号最后一位十进制数字，指出 main 函数中变量 I 在每条赋值语句执行后的值（每小题 2.5 分，共 15 分）

```
int x = _____                    //学号最后一位十进制数
int y = x + 3;
struct A {
    int x;
    static int &y;
public:
    operator int( ) const { return x + y; }
    int &v(int &x) {
        for(int y = 1; x < 301; x ^= y++) {
            if(x > 300) { x -= 31; y -= 2;
            }
            return ++x;
        }
        A &operator++( ) { ++x; ++y; return *this; }
        A(int x = A::y + 2, int y = ::x + A::y) { A::x = x + 1; A::y = y + 2; }
};
int &A::y = ::y;
void main( ) {
    A a(3, 4), b(a), c;
    int i, &j = i;
    int A::*p = &A::x;
    j = a.x;                     // i =
    i = a.y;                     // i =
    i = a.*p;                    // i =
    i = ++a;                     // i =
    i = b.y + ::y;               // i =
    (b.v(i) = 3) += 2;           // i =
}
```

答案（假设学号最后一位 = n）：

```
void main( ) {
    A a(3,4);                    //a.x=4, ::y=6
    A b(a);                      //b(a) 浅拷贝构造:  b.x=4, ::y=6
    A c;                         //c(::y+2,::x+::y)=c(8, n+6), c.x=9, ::y=n+8
    int i, &j = i;
    int A::*p = &A::x;
    j = a.x;                     //i=4
    i = a.y;                     //i=n+8
```

```
    i = a.*p;                   //i=4
    i = ++a;                    //++a: a.x=5,::y=n+9, i=n+14
    i = b.y + ::y;              //i=2n+18
    (b.v(i) = 3) += 2;          //i=5
};
```

分析：

	::x	::y, A::y	a.x	b.x	c.x	i, j
	n	n+3				
a(3, 4)		6	4			
b(a)		6		4		
c <=> c(8, n+6)		n+8			9	
j = a.x						4
i = a.y						n+8
i = a.*p						4
i = ++a		n+9	5			n+14
i = b.y + ::y						2n+18
(b.v(i) = 3) += 2						5

六、一个自然数，如果它等于除其本身之外的所有其它不同因子之和，则这个自然数被称为完美数。例如，6=1+2+3, 28=1+2+4+7+14。试编写如下完美数类中的所有函数成员的函数体代码（每小题 2.5 分，共15 分）

```
class PER {
    const int n;            //存放自然数
    int *const f;           //存放所有有效因子, 所有因子之和等于 n 才是完美数
    int c;                  //有效因子个数：正数表示是完美数，负数表示不是
public:
    PER(int p);                         //用自然数 p 初始化 n, f, c
    PER(const PER &p);                  //深拷贝构造函数
    PER &operator=(const PER &p);       //深拷贝赋值运算
    operator int ( ) const;             //若不是完全数则返回 0，否则返回 c
    int operator [ ] (int k) const;     //返回 k 所指示的因子, 若 k < 0 或 k >= c 返回 0
    ~PER( );                            //析构函数
};
//提示: f 分配的整型内存单元数量不会超过 n / 2。
```

PER::PER(int p): n(p), f(new int [p])

```
{
        c = 0;
        for(int k = 1; k < n; k++)
        {
                if((n % k) == 0) f[c++] = k;
        }
        int sum = 0;
        for(int k = 0; k < c; k++) sum += f[k];
        if(sum != n) c = -1;
}

PER::~PER()
{  if(f) { delete[] f; *(int *)&f = 0;  }  }

//深拷贝构造函数
PER::PER(const PER &p): n(0),f(0)
{  *this = p;  }

//深拷贝赋值运算
PER &PER::operator=(const PER &p)
{
        this->~PER();
        *(int *)&n = p.n;
        *(int **)&f = new int[n];
        c = p.c;
        for(int k = 0; k < c; k++) f[k] = p.f[k];
        return *this;
}

//若不是完全数则返回 0，否则返回 c
PER::operator int( ) const
{  return c > 0 ? c : 0;  }

//返回 k 所指示的因子, 若 k < 0 或 k >= c 返回 0
int PER::operator[](int k) const
{  return k < 0 || k >= c ? 0 : f[k];  }
```

面向对象程序设计模拟试卷

一. 单选题(15)。

1. 关于定义“struct A{int x; const int y=3;};volatile A a={1};”，如下叙述那个__D___正确:
   A. a.x 是 int、a.y 是 const int 类型　　B. a.x 是 volatile int、a.y 是 const int 类型
   C. a.x 是 int、a.y 是 const volatile int 类型 D. a.x 是 volatile int、a.y 是 const volatile int 类型

2. 关于 inline constexpr int &f(int &&a)的调用叙述__B___正确:
   A.该用常量做实参调用，返回传统右值　　B. 该用常量做实参调用，返回传统左值
   C.可用变量做实参调用，返回传统右值　　D. 可用变量做实参调用，返回传统左值

3. 若派生类函数不是基类的友元，关于该函数访问基类成员___C__正确:
   A. 只有公有的可被派生类函数访问　　B. 都可以被派生类函数访问
   C. 公有和保护的可被派生类函数访问　　D. 都不对

4. 关于函数的所有缺省参数的叙述__B___正确:
   A. 只能出现在参数表的最左边　　B. 只能出现在参数表的最右边
   C. 必须用非缺省的参数隔开　　D. 都不对

5. 对于定义“char * &&f( );”，如下哪个语句是错误的__A___:
   A. f( )=(char*) "abcd";　　B. *f( )= 'A';
   C. char *p=f( );　　D. *f( )= "ABC"[1];

二. 在最多使用一级作用域“::”访问如 A::c 的情况下，指出各类可访问的成员及其访问权限(20) 。

```
class A{
        int    a;
protected:
        int    b;
public:
        int    c;
};
class B: protected A{
        int    d;
protected:
        int    e;
public:
       A::c;
        int    f;
};
```

```
class C:    A{
        int    g;
protected:
        int    h;
public:
        int    i;
};
class D: B, C{
        int    j;
protected:
       B::b;
        int    k;
private:
        int    n;
};
```

类 A 的可访问成员：

private:　　int a;
protected:　　int b;
public:　　int c;

类 B 的可访问成员：

private:　　int d;
protected:　　int b(或 A::b),e;
public:　　int c(或 A::c),f;

类 C 的可访问成员：

private: int b(或 A::b), c(或 A::c),g;

protected: int h;

public: int i;

类 D 的可访问成员：

private: int c(或 B::c),e(或 B::e), f(或 B::f),h(或 C::h), i(或 C::i),j,n;

protected: int b(或 B::b),k;

三. 指出 main 中每行的输出结果(20) 。

```
#include <iostream.h>
struct A{A( ){ cout<<'A';}};
struct B{B( ){ cout<<'B';}};
struct C: A{C( ){ cout<<'C';}};
struct D: virtual B, C{D( ){ cout<<'D';}};
struct E: A{
      C   c;
      E( ): c( ){ cout<<'E';}
};
struct F: virtual B, C, D, E{
     F( ){ cout<<'F';}
};
void main( ){
      A   a;   cout<<'\n';        //输出=A
      B   b;   cout<<'\n';        //输出=B
      C   c;   cout<<'\n';        //输出=AC
      D   d;   cout<<'\n';        //输出=BACD
      E   e;   cout<<'\n';        //输出=AACE
      F   f;   cout<<'\n';        //输出=BACACDAACEF
}
```

四. 指出以下程序的语法错误及其原因(15) 。

```
class A {
    static int  a=0;//(1)前面有inline或const或constexpr才能初始化,否则在类外初始化
protected:
    int  b;
public:
    int  c;
    A(int) {};
    inline constexpr operator int() { return b; };
}  a(1, 2);   //(2)没有两个参数的构造函数
class B: A {
    B(int m) { b = m; }; //(3)A不存在无参构造函数供“B(int m):”后调用
    using A::b;
    virtual int d;//(4)virtual不能用于说明数据成员
```

```
    int  e;
public:
    int b;
    friend B& operator =(const B& b){return *this;}//(5)等号仅单参，应为实例函数
    static B(int, int); //(6)不能用static定义构造函数
} b = 5; //(7)单参构造函数是私有的，无法访问
class C: B {
public:
    C operator++(double) { return *this; };//（8）后置运算必须int类型定义显式参数
}c;  //(9)类C无法生成无参构造函数初始化c
int main( ) {
    int* A::* p, i;
    i = a.a;  //(10)私有成员，main函数无法访问
    i = A(4);
    i = b.c;  //（11）A继承到B的成员B::c是私有的，main函数无法访问
    p = &A::c;//(12)&A::c的类型为int A::*，与p的类型int* A::*不同，不能赋值给p
    i = b;//(13)B未定义operator int函数，继承自A的operator int私有化，main不能访问
    return;//(14)main要求返回一个整型值
 }
```

五. 指出 main 变量 i 在每条赋值语句执行后的值(15) 。

```
int   x = _____(请填入本人学号最后一位数字),   y = x + 30;
struct A {
      static int   x;
      int   y;
public:
      operator int( ){ return x-y; }
      A operator ++(int){ return A(x++, y++); }
      A(int x=::x+2, int y=::y+3){ A::x=x;   A::y=y; }
      int &h(int &x);
};
int &A::h(int &x)
{
    for(int y=1; y!=1|| x<201; x+=11, y++)   if(x>200) { x-=21; y-=2;}
      return x-=10;
}
int A::x = 23;
void main( ) {
      A   a(54, 3),   b(65),   c;
      int   i,   &z=i,   A::*p=&A::y;
      z=b.x;
      i=a.x;
      i=c.*p;
      i=a++;
```

```
    i=::x+c.y;
    i=a+b;
    b.h(i)=7;
}
```

答案：

学号尾数	z = b.x	i = a.x	i=c.*p	i= a++	i=::x+c.y	i=a+b	b.h(i)=7
0	2	2	33	-1	33	-33	7
1	3	3	34	0	35	-32	7
2	4	4	35	1	37	-31	7
3	5	5	36	2	39	-30	7
4	6	6	37	3	41	-29	7
5	7	7	38	4	43	-28	7
6	8	8	39	5	45	-27	7
7	9	9	40	6	47	-26	7
8	10	10	41	7	49	-25	7
9	11	11	42	8	51	-24	7

分析：

	::x	::y	A::x	a.y	b.y	c.y	i, z
	n	n+30	23				
a(54, 3)			54	3			
b(65) <=> b(65, n+33)			65		n+33		
c <=> c(n+2, n+33)			n+2			n+33	
z=b.x							n+2
i=a.x							n+2
i=c.*p							n+33
i = a++			n+2	4			n-1
i=::x+c.y							2n+33
i=a+b							n-33
b.h(i)=7							7

六. 为了没有误差地表示分数，定义分数类 FRACTION 用来表示分数，用整型 numerator 存分子、整型 denominator 存分母；并用*重载分数约简运算、用>重载分数比较运算、用+重载分数加法运算、用*重载分数乘法运算，以及相关的构造函数；可运用最大公约数函数 cmd 约简 (15)。

```
int    cmd(int x, int y){
       int r;
       if(x<y){    r=x;    x=y;    y=r;    }
       while(y!=0){    y=x%(r=y);    x=r; }
       return x;
}
```

解：

```
class FRACTION{ //对于6/7，numerator 存分子 6，denominator 存分母 7
       int      numerator, denominator;
public:
       int operator>(const FRACTION&)const;              //大于比较，例 6/7>2/3
       FRACTION(int num, int den=1);                     //num、den 各为分子和分母
       FRACTION operator*( )const;                       //分数约简，*30/36=5/6
       FRACTION operator+(const FRACTION&)const;         //加法，6/7+2/3=32/21
       FRACTION operator*(const FRACTION&)const;         //乘法，6/7*2/3=12/21=4/7
};

FRACTION::FRACTION(int num, int den){
      numerator=num;
      denominator=den;
}
int FRACTION::operator>(const FRACTION&f)const{
     double d= denominator *f.denominator;
      return    numerator*f.denominator/d > denominator*f.numerator/d;
}
FRACTION FRACTION::operator*( )const{
      int    c=cmd(numerator, denominator);
      return FRACTION(numerator/c, denominator/c);
}
FRACTION FRACTION::operator+(const FRACTION&f)const{
      int    n= numerator*f.denominator+denominator*f.numerator;
      int    d= denominator*f.denominator;
      return *FRACTION(n, d);      //对运算结果进行约分运算
}
FRACTION FRACTION::operator*(const FRACTION&f)const{
      return *FRACTION(numerator*f.numerator,    denominator*f.denominator); //约分
}
```

华中科技大学计算机科学与技术学院 2020~2021 第一学期

“C++程序设计”考试试卷（A 卷）

考试方式 开卷 考试日期 2020-11-14 考试时长 150 分钟

专业班级 _____ 学 号 _____ 姓 名 _____

题号	一	二	三	四	五	六	总分	核对人
分值	15	20	20	15	15	15	100	
得分								

分 数	
评卷人	

一、单选题：请从 4 个选项中选择一个最合适的选项作为答案（15 分：每小题 3 分）。

1. 关于定义“struct A{int x; mutable int y;}const a={1,3};”，如下叙述哪个__B___正确:
   A. a.x 可被赋值，a.y 不可被赋值　　B. a.x 不可被赋值，a.y 可被赋值
   C. a.x 和 a.y 均不可被赋值　　D. a.x 和 a.y 均可被赋值
2. 关于 union 定义的类的叙述___A__正确:
   A. 既不能是基类也不能是派生类　　B.不能是基类，但可以是派生类
   C. 可以是基类，但不能是派生类　　D.既可以是基类，也可以是派生类
3. 对于说明“int &f( ); int &&g( );”及其函数调用 f( )和 g( )，如下 4 个叙述___D__正确:
   A. 调用 f( )和调用 g( )均不可被赋值　　B.调用 f( )不可被赋值，调用 g( )可被赋值
   C. 调用 f( )和调用 g( )均可被赋值　　D.调用 f( )可被赋值，调用 g( )不可被赋值
4. 对于定义“struct A{ int f( ) ;}”，关于 int 前面是否可用 static 和 virtual，如下的叙述__C___错误:
   A. 可以只使用 static　　B. 可以只使用 virtual
   C. 必须同时使用 static 和 virtual　　D. 都不对(错)
5. 对于定义“char *const &f( );”，如下哪个语句是错误的___A__:
   A. f( )=(char*) "abcd";　　B. *f( )= 'A';
   C. char *p=f( );　　D. *f( )= "ABC"[1];

分 数	
评卷人	

二、在最多使用单重作用域例如 A::x 的前提下，在空白处填写以下各类可被访问的成员及其访问权限（20 分：根据正确回答的成员个数按比例计算给分）。

```
class A{              //类 A 的可访问成员：
    int a;            //私有成员：a;
protected:            //保护成员：b, c;
    int b, c;         //公有成员：d, e;
```

```
public:
    int d, e;
};
class B: protected A {//类 B 的可访问成员：
    int a;              //私有成员：a;
protected:              //私有成员：
    int b, f;           //保护成员：b, f;  A::(b, c, d, e);
    using A::d;         //保护成员：
public:                 //公有成员：e, g;
    int e, g;           //公有成员：
};
struct C: A {           //类 C 的可访问成员：
    int a;              //私有成员：
protected:              //私有成员：
    int b, f;           //保护成员：b, f;  A::(b, c);
public:                 //保护成员：
    int e, g;           //公有成员：a, e, g;  A::(d, e);
    using A::d;         //公有成员：
};
struct D: B, C {        //类 D 的可访问成员：
    int a;              //私有成员：
protected:              //私有成员：
    int b, f;           //保护成员：b, f;  B::(b, f;  A::(b, c, d, e));  C::(b, f;  A::(b, c))
public:                 //保护成员：
    int e, g;           //公有成员：a, e, g;  B::(e, g);  C::(a, e, g;  A::(d, e))
};                      //公有成员：
```

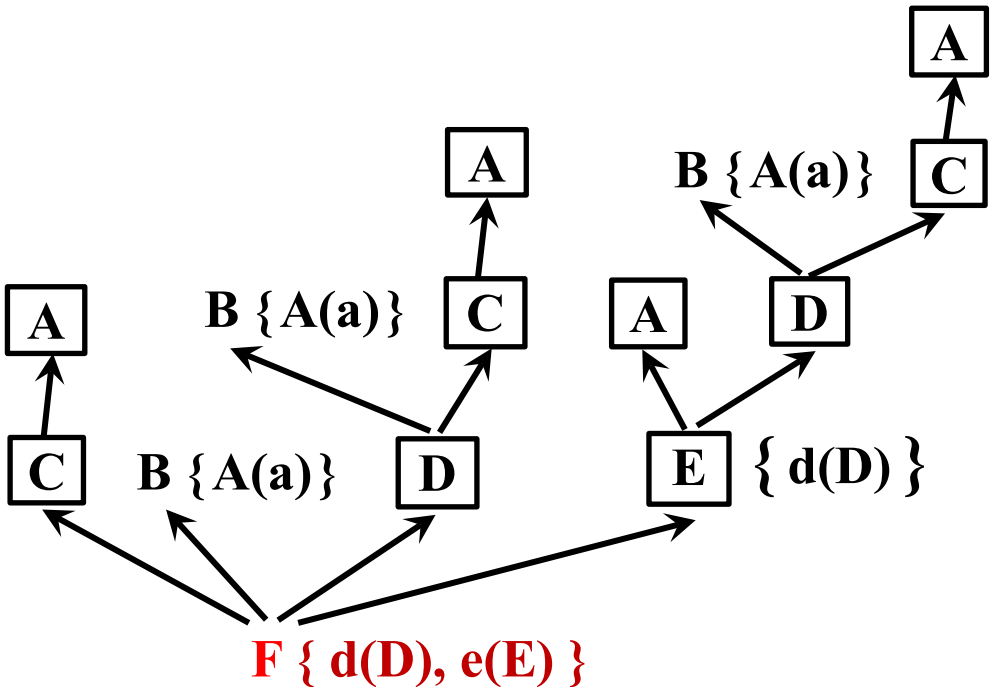
分　数	
评卷人	

三、回答 main 中每行语句的输出结果（20 分：前四个语句的输出每个 3 分，后两个语句的输出每个 4 分）。

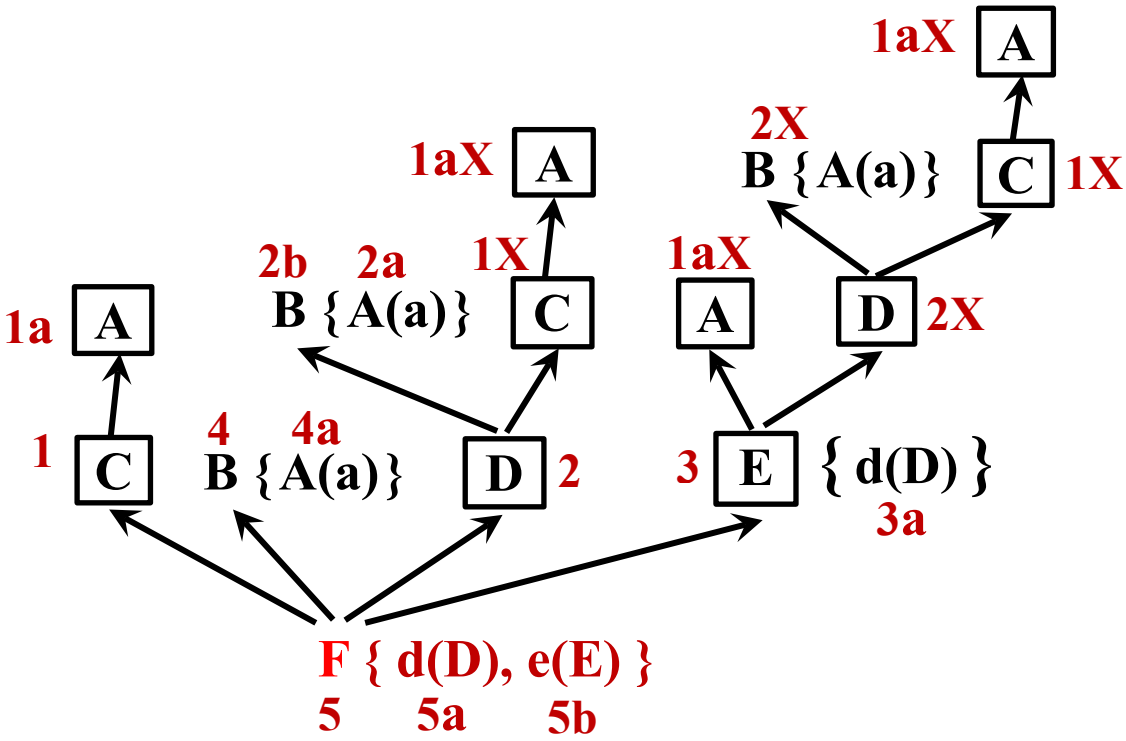
```
#include <iostream>
using namespace std;
struct A { A( ) { cout << 'A'; } };
struct B { A a; B( ) { cout << 'B'; } };
struct C : virtual A { C( ) { cout << 'C'; } };
struct D : B, virtual C { D( ) { cout << 'D'; } };
struct E : virtual A, virtual D {
    D  d;
    E( ) : A( ) { cout << 'E'; }
};
struct F : virtual C, B, virtual D, virtual E {
    D  d;  E  e;
```

```
    F( ) { cout << 'F'; }
};
void main( ) {
    A   a;   cout << '\n';          //输出=A
    B   b;   cout << '\n';          //输出=AB
    C   c;   cout << '\n';          //输出=AC
    D   d;   cout << '\n';          //输出=ACABD
    E   e;   cout << '\n';          //输出=ACABDACABDE
    F   f;   cout << '\n';          //输出=ACABDACABDEABACABDACABDEF
}
```

F f 分析如下：



解答内容不得超过装订线



D: ACABD

E: ACABD ACABD E

F: AC ABD ACABDE AB ACABD ACABDE F

分　数	
评卷人	

四、综合分析并指出以下程序中下划线位置可能出现的语法错误及其原因 (共 15 分：每错约 1 分)。

```
class A {
      int   a;
protected:
      virtual ~A( ) { }
public:
      int& b;
      int   c;
      virtual A(*g)( )=0;          //错误1：不能用virtual定义数据成员_____
      virtual A(int x)__{ a = x; };  //错误2：构造函数不能用virtual定义_错误3：没有初始化b____
} a = (4, 3);                      //错误4：不能调用保护的析构函数_____
class B: A {
      int   d;
      using A::a;                  //错误5：私有a不可访问_____
public:
```

```
    friend int operator( )(int) { return 2; }; //错误6：（）定义为非成员函数的友元_____
    B(int x, int y, int z)____{ d = x + y + z; }; //错误7：未初始化基类_____
} b(5, 6, 7);
class C: B {
    int z;
public:
    ~C(int x) { z = x; };          //错误8：不应有参数_____
} c ;                              //错误9：无法生成无参构造函数_____
void  main( ) {
    int  A::* p = &A::b;           //错误10：无法取得引用成员的地址_____
    int  i = a.a;                  //错误11：无法访问私有成员_____
    int&& y = i ;                  //错误12：无址引用不能引用有址变量_____
    i = b.b;                       //错误13：无法访问私有的 B::b_____
    i = i + c.d;                   //错误14：不存在 c.d_____
    i = b.*p;                      //错误15：B非A的子类，不能使用实例成员指针int A::*_____
}
```

解答内容不得超过装订线

分　数	
评卷人	

五、请填入自己学号的最后一位十进制数字，计算 main 函数中变量 i 在每条赋值语句执行后的值 (共 15 分：每小题 2.5 分)。

```
int  x = 填写自己学号最后一位十进制数, y = x + 3;
struct A {
    int x = ::x + 2;
    static int &y;
public:
    operator int( )const { return x + y; }
    int& v(int& x) {
        for (int y = 1; x < 301; x ^= y, y++)
            if (x > 300) { x -= 31; y -= 2; }
        return ++x;
    }
    A& operator++( ) { ++x; ++y; return *this; }
    A(int x, int y = ::y + 3) { A::y = y; }
};
int &A::y = ::y;
void main( ) {
    A   a(2, 7), b(5);
    int  i, &j = i, A::*p = &A::x;
    i = a.y;                       //i=
    j = a.x;                       //i=
    i = a.*p;                      //i=
    i = ++a;                       //i=
```

```
    i = b.y + ::y;                  //i=
    (b.v(i) = 5) += 2;              //i=
}
```

答案：

	i = a.y	j = a.x	i = a.*p	i = ++a	i = b.y + ::y	(b.v(i) = 5) += 2
0	10	2	2	14	22	7
1	10	3	3	15	22	7
2	10	4	4	16	22	7
3	10	5	5	17	22	7
4	10	6	6	18	22	7
5	10	7	7	19	22	7
6	10	8	8	20	22	7
7	10	9	9	21	22	7
8	10	10	10	22	22	7
9	10	11	11	23	22	7

分析：

	::x	**::y, A::y**	**a.x**	**b.x**	**i**
	n	**n+3**			
a(2, 7)		**7**	**n+2**		
b(5)		**10**		**n+2**	
i = a.y					**10**
j = a.x					**n+2**
i = a.*p					**n+2**
i = ++a		**11**	**n+3**		**n+14**
i = b.y + ::y					**22**
(b.v(i)=5) += 2					**7**

分　数	
评卷人	

六、对于如下所有单词按升序排列的字典类 DCT，对其中的函数成员进行程序设计(共 15 分：每个函数 1.5 分)。

```
class DIC {
    char** const e;                 //用于存放字典的单词
    const int m;                    //能够存放的单词个数
    int r;                          //已经存放的单词个数
public:
    DIC(int m=1000);                //创建最多存 m 个单词的字典，所有 e[i]置空指针
```

```
    DIC(const DIC& d);                          //根据已知字典 d 深拷贝构造新字典
    DIC(DIC&& d)noexcept;                       //根据已知字典 d 移动构造新字典
    DIC& operator=(const DIC& d);               //深拷贝赋值运算符的重载
    DIC& operator=(DIC&& d)noexcept;            //移动赋值运算符的重载
    DIC& operator<<(const char* w);             //将单词插入字典并保持升序，若有该单词则不插
    operator int( )const noexcept;              //获得字典实际存放的单词个数
    const char* operator[ ](int i)const noexcept;  //获得下标为 i 的单词
    int operator( )(const char* w)const noexcept;  //查找单词 w 在字典中的位置
    ~DIC( )noexcept;                            //析构字典
};
```

答案：

```
#define _CRT_SECURE_NO_WARNINGS
#include <string.h>
DIC::DIC(int m) :e(new char* [m] {}), m(e ? m : 0), r(0) { };
DIC::DIC(const DIC& d) :e(new char* [d.m]{}), m(e ? d.m : 0), r(d.r) {
    if (e) throw "memory not enough!";
    for (int h = 0; h < r; h++) {
        e[h] = new char[strlen(d.e[h]) + 1];
        if (e[h] == nullptr) throw "memory not enough!";
        strcpy(e[h], d.e[h]);
    }
};
DIC::DIC(DIC&& d)noexcept : e(d.e), m(d.m), r(d.r) {
    (char**&)(d.e) = nullptr;
    (int&)(d.m) = r = 0;
}
DIC& DIC::operator=(const DIC& d) {
    if (this == &d) return *this;
    if (e) {
        for (int h = 0; h < r; h++) {
            if (e[h]) delete [ ] e[h];
        }
        delete [ ]e;
    }
    (char**&)e = new char* [d.m];
    if (e== nullptr) throw "memory not enough!";
    (int&)m = d.m;
    r = d.r;
    for (int h = 0; h <r; h++) {
        e[h] = new char[strlen(d.e[h]) + 1];
        if (e[h] == nullptr) throw "memory not enough!";
        strcpy(e[h], d.e[h]);
    }
    return *this;
}
DIC& DIC::operator=(DIC&& d)noexcept {
    if (this == &d) return *this;
```

解答内容不得超过装订线

```
    if (e) {
        for (int h = 0; h < r; h++) {
            if (e[h]) delete[ ] e[h];
        }
        delete[]e;
    }
    (char**&)e=d.e;
    (int&)m = d.m;
    r = d.r;
    (char**&)(d.e) = nullptr;
    (int&)(d.m) = r = 0;
    return *this;
}
DIC& DIC::operator<<(const char* w) {
    if (operator( )(w) != -1) return *this;
    int h;
    if (r == m) throw "memory not enough for insert!";
    for (h = 0; h < r; h++)
        if (strcmp(e[h], w) > 0) {
            for (int x = r; x > h; x--) e[x]=e[x - 1];
            e[h] = new char[strlen(w) + 1];
            if (e[h] == nullptr) throw "memory not enough!";
            strcpy(e[h], w);
            r++;
            break;
        }
    return *this;
}
DIC::operator int( ) const noexcept { return r; }
const char* DIC::operator[ ](int i)const noexcept {
    if (i<0 || i>=r) throw "subscription overflowed!";
    return e[i];
}
int DIC::operator( )(const char* w)const noexcept {
    for (int h = 0; h < r; h++)
        if (strcmp(e[h], w) == 0) return h;
    return -1;
}
DIC::~DIC( ) noexcept {
    if (e) {
        for (int h = 0; h < r; h++) {
            if (e[h]) delete[ ] e[h];
        }
        delete [ ]e;
        (char**&)e = nullptr;
        (int&)m = r = 0;
    }
}
```