第一题

```
void f() {
    c=cc;
    //1.编译报错，不能将const char赋值给普通的char
    cc=c;
    //2.编译通过，但运行时错误:cc是const char，虽然允许读取c的值，
    //但在运行时会因为尝试修改const对象报错
    pcc=&c;
    //3.编译通过，运行正确:pcc是指向const char的指针，可以指向char
    pcc = &cc;
    //4.编译通过，运行正确:pcc是指向const char的指针，cc是const char
    pc=&c;
    //5.编译通过，运行正确:pc是指向char的指针，c是char，匹配
    pc=&cc;
    //6.编译报错:pc是指向char的指针，不能指向const char
    pc = pcc;
    //7.编译报错:不能将const char*赋值给char*
    pc = cpcc;
    //8.编译报错:不能将const char*赋值给char*
    cpcc = pc;
    //9.编译报错:cpcc是const char* const，常量指针无法修改
    *pc = "ABCD"[2];
    //10.编译通过，但运行时错误:可以将字符C赋值给*pc，但此时pc为c空指针
    cc = a;
    //11.编译报错:cc是const char，无法修改
    *cpcc = *pc;
    //12.编译报错:cpcc是const char* const，不能通过它修改指向的值
    pc = *pcpc;
    //13.编译通过，但运行时错误:*pcpc是指向char*的指针，类型匹配，
    //但此时pcpc是悬挂指针
    **pcpc = *pc;
    //14.编译通过，但运行时错误:*pcpc解引用得到char*，**pcpc是char
    //可以赋值，但此时pcpc是悬挂指针，pc为空指针
    *pc = **pcpc;
    //15.编译通过，但运行时错误:**pcpc是char，*pc也是char，
    //可以赋值，但此时pcpc是悬挂指针，pc为空指针
    *pcc = 'b';
    //16.编译报错:pcc是指向const char的指针，不能修改*pcc
    *pcpc = 'c';
    //17.编译报错:*pcpc是char*，不能直接赋值为char，需要指向一个地址
    *cpcc = 'd';
    //18.编译报错:cpcc是const char* const，不能修改*cpcc
    *pcpc = pc;
    //19.编译通过，运行正确:*pcpc是指向char*的指针，可以指向pc
    pcpc = &cpc;
    //20.编译通过，运行正确:&cpc是char* const*，可以赋给char* const*
    pccp = &cpc;
    //21.编译通过，运行正确:pccp是char* const* const，
    //可以指向char* const*
}
```

第二题

```
int* (*p)[4];
//p是一个指向数组的指针，数组包含4个元素，每个元素是一个指向int的指针

int* (*t)(int) = 0;
typedef int* (*F)(int);
F f = t;
//t是一个函数指针，指向一个函数，该函数的参数是int类型，返回指向int
//的指针，t初始化为空；typedef定义了一个函数指针类型F，它代表一个函
//数指针，指向的函数参数为int类型，返回 指向int的指针；f和t都是函数
//指针，指向接受一个int参数并返回int*的函数

int (*(*g)[10])(int, int) = 0;
typedef int (*G[10])(int, int);
G *pg = g;
//g是一个指向包含10个元素的数组的指针，数组中的每个元素是一个函数指
//针，指向接受两个int参数并返回int的函数；typedef定义了一个类型G，
//表示一个包含10个函数指针的数组，这些函数指针指向接受两个int参数并
//返回int的函数；pg和g都是指向函数指针数组的指针，指向的是包含10个
//函数指针的数组
```

第三题

```
int& ri_1 = i;
//1.编译通过，ri_1是对i的引用，i是一个普通的int，可以引用
int& ri_2 = ci;
//2.编译报错，ci是const int，而ri_2是一个普通的int&，不能引用 const变量
int& ri_3 = d;
//3.编译报错，d是double类型，不能用int&来引用double类型的变量
int& ri_4 = i * 2;
//4.编译报错，i*2是一个右值表达式，不能绑定到int&（左值引用）
int& ri_5;
//5.编译报错，引用变量必须初始化
int& ri_6 = i++;
//6.编译报错，i++是一个右值表达式，不能绑定到int&（左值引用）
int& ri_7 = --i;
//7.编译通过，--i是一个左值表达式，可以绑定到int&（左值引用）
int& ri_8 = f(i);
//8.编译报错，f(i)返回的是一个右值（临时值），不能绑定到int&
int& ri_9 = g(i);
//9.编译通过，g(i)返回一个左值引用（int&），可以绑定到ri_9

const int& cri_1 = i;
//10.编译通过，i是一个普通的int，可以绑定到const int&
const int& cri_2 = 1;
//11.编译通过，临时值（右值）可以绑定到const int&
const int& cri_3 = d;
//12.编译通过，临时值类型的double可以绑定到const int&。它会进行类型转换，
```

```
//将d转换为int后再绑定到const int&
const int& cri_4 = 3.14;
//13.编译通过，3.14是一个double，会被转换为int，然后绑定到const int&
const int& cri_5 = f();
//14.编译报错，这里f()没有参数

int& &rri_1 = 1;
//15.编译报错，不能定义引用的引用
int& *pri;
//16.编译错误，指针不能指向引用变量，引用变量无地址
int&& rri_2 = 1;
//17.编译通过，1是一个右值，可以绑定到右值引用int&&
int&& rri_3 = i;
//18.编译报错，i是左值，不能绑定到右值引用int&&，右值引用只能绑定右值
int&& rri_4 = f(i);
//19.编译通过，f(i)返回的是右值，可以绑定到右值引用int&&
int&& rii_5 = rri_4;
//20.编译通过，rri_4是一个右值引用，可以绑定到另一个右值引用rri_5
int& ri_10 = rri_4;
//21.编译报错，rri_4是右值引用，不能绑定到左值引用int&，左值引用只能绑定左值
```

第四题

```
void test() {
    f1(a);
    //1.成立，a是一个数组名，在表达式中会被转换为指向其第一个元素的指针（int*）
    f2(a);
    //2.成立，f2接受的是int*类型的引用，a 在这里转换为int*，可以绑定到int*&引用
    f3(a);
    //3.成立，f3接受的是指向int的常量指针的引用，虽然a转换为int*，但int*可以绑定
    //到int* const&，因为const&可以引用非const对象
    f4(a);
    //4.成立，f4接受的是一个对大小为3的int数组的引用，a是一个大小为3的int数组
}
```

第五题

```
int& f(int &x){
    for (int y = 1; y != 1 && x < 50; x += 13, y++)
        if (x > 49) {
            x -= 31;
            y -= 5;
        }
    return x -= 20;
}
void test() {
    int i = 40;
    f(i) = 1;
}
```

```
//执行完test()函数后，i的值为1
//调用f(i)，进入f函数
//x是i的引用，因此x的初始值为40，y=1，然后进入for循环
//由于y=1，y!=1是false，因此整个for循环不会执行
//执行x-=20，x减去20，变为 20。
//函数返回的值是x的引用
//f(i)返回i的引用，因此f(i)=1是在将i赋值为1
```

第六题

```
void swap(int x, int y) {
    //交互x、y的值
    int temp = x;
    x = y;
    y = temp;
}
void test() {
    int i = 0, j = 1;
    swap(i, j);
}
//不互换，因为传入swap()函数的参数是形参，形参的改变不会影响实参的值

//修改如下：
void swap(int &x, int &y) {
    int temp = x;
    x = y;
    y = temp;
}
```

第七题

```
//请实现如下函数创建如下的不规则数组，并打印在屏幕上
0 0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0
0 0 0 0
0 0 0
0 0
0
```

```
//参数size是数组第一行的列数，后面每一行列数递减1，直到最后一行只有一个元素。
void matrix(int size) {
    // 创建不规则数组，分配每一行的列数不同的空间
    int** array = (int**)malloc(size * sizeof(int*));
    for (int i = 0; i < size; ++i) {
        array[i] = (int*)malloc((size - i) * sizeof(int));
```

```
            // 初始化数组元素
            for (int j = 0; j < size - i; ++j) {
                array[i][j] = 0;
            }
        }
        // 打印数组内容
        for (int i = 0; i < size; ++i) {
            for (int j = 0; j < size - i; ++j) {
                std::cout << array[i][j] << " ";
            }
            std::cout << std::endl;
        }
        // 释放动态分配的内存
        for (int i = 0; i < rows; ++i) {
            free(array[i]);
        }
        free(array);
    }
```

更正

一:

1 编译正确运行也正确，char = const char没有问题

2 编译错误，cc是const的，不能再被赋值，即不能出现在等号左边

19 编译错误，*pcpc是char * const类型的一级指针，

20 编译正确运行也正确，pcpc本身不是const

21 编译错误，pccp本身是const的，不能被赋值

三:

14 编译通过，const左值引用可以用右值表达式f(i)来初始化

20 编译报错，rri_4是右值引用，但是是左值，不能绑定到右值引用rri_5

21 编译不报错，非const左值引用用求值结果相同的左值表达式rri_4初始化

四:

2 不可以，a类型为int [3]，实参传递给形参相当于int *&pr = a; pr是非const左值引用，不能用不同类型的左值来绑定