问题一

1.区分成员变量与参数：在构造函数或成员函数中，如果参数名称与成员变量相同，可以通过`this`指针来明确访问成员变量

2.返回当前对象：在链式调用中，`this`指针可以返回当前对象的引用，允许连续调用多个成员函数。

3.支持多态：在基类的虚函数中，`this`指针可以确保调用的是派生类的函数，实现多态性。

4.常成员函数的使用：在常成员函数中，`this`指针的类型为`const MyClass*`，表示不能修改对象的成员变量。

问题二

什么情况需要析构函数：

管理动态分配的内存：如果类中有通过`new`动态分配的内存，必须在析构函数中使用`delete`释放它，以防止内存泄漏。

关闭文件或网络连接：如果类中打开了文件或网络连接，析构函数中应该负责关闭这些连接，以释放系统资源。

释放其他资源：例如，数据库连接、线程或任何其他需要手动管理的资源。

释放的资源包括：

动态分配的内存（使用`new`分配的内存）；打开的文件句柄；网络连接；其他任何需要显式释放的系统资源。

不包括：

静态或全局资源：如静态变量或全局对象，这些资源的生命周期不受类的实例控制。

栈上分配的资源：局部变量（在栈上分配的资源）在离开作用域时会自动被释放，无需在析构函数中处理。

对象的成员变量：如果成员变量是其他类的对象，C++会自动调用这些对象的析构函数，负责其自身资源的释放。

问题三

在哪些情况下会调用拷贝构造函数：

1.对象作为值传递给函数时

2.对象作为函数返回值时

3.用一个对象初始化另一个对象的时候

为什么隐式拷贝构造函数可能导致运行错误：

隐式拷贝构造函数是浅拷贝：两个对象会共享同一块动态内存，若一个对象被析构，另一个对象的指针将指向已经释放的内存，导致悬空指针，并且若两个对象分别释放同一块内存，可能会导致程序崩溃或未定义行为。

问题四

浅拷贝两个对象会共享同一块动态内存，指向同一个空间，一个对象被析构，另一个对象的指针将指向已经释放的内存，而深拷贝是另外分配空间，将相同的值拷贝，与原对象互不影响。

问题五

```cpp
#include <iostream>
#include <string>

using namespace std;
class Accout {
public:
    Accout(string name = string(), string accout = string(), int deposit = 0)
        :_name(name)
        ,_accout(accout)
        ,_deposit(deposit)
    {}
    void show_information() {
        //显示账户信息
        cout << "name:" << _name << endl;
        cout << "accout:" << _accout << endl;
        cout << "deposit:" << _deposit << endl;
        return;
    }
    void save_in(size_t num_money) {
        //存款
        _deposit += num_money;
        return;
    }
    void withdraw(size_t num_money) {
        //取款
        if (_deposit >= num_money) {
            _deposit -= num_money;
        }
        else {
            cout << "Insufficient balance" << endl;
        }
        return;
    }
private:
    //成员
    string _name;
    string _accout;
    int _deposit;
};

int main() {
    Accout acc("kiki", "231880038", 100);
    acc.show_information();
    acc.withdraw(200);
    acc.save_in(500);
    acc.withdraw(200);
    return 0;
}
```

问题六

```cpp
#include <iostream>

using namespace std;

struct node {
    int val;
    node* next;
};
class Deque {
public:
    Deque()
        :_head(new node())
    {
        _head->next = nullptr;
    }
    ~Deque() {
        node* cur = _head;
        while (cur) {
```

```
            node* temp = cur;
            cur = cur->next;
            delete temp;
            temp = nullptr;
        }
    }
    void push_front(int x) // 在队列前端入队一个元素x
    {
        node* newnode = new node();
        newnode->val = x;
        newnode->next = _head->next;
        _head->next = newnode;
    }
    void push_back(int x) // 在队列后端入队一个元素x
    {
        node* cur = _head;
        while (cur->next) {
            cur = cur->next;
        }
        node* newnode = new node();
        newnode->val = x;
        newnode->next = nullptr;
        cur->next = newnode;
    }
    void pop_front() // 在队列前端出队一个元素
    {
        if (_head->next) {
            //不为空
            node* temp = _head->next;
            _head->next = temp->next;
            delete temp;
            temp = nullptr;
        }
        else {
            return;
        }
    }
    void pop_back() // 在队列后端出队一个元素
    {
        if (_head->next) {
            //不为空
            node* cur = _head;
            node* pre = nullptr;
            while (cur->next) {
                pre = cur;
                cur = cur->next;
            }
            pre->next = nullptr;
            delete cur;
            cur = nullptr;
        }
        else {
            return;
        }
    }
    size_t size() const // 返回队列中元素个数
    {
        node* cur = _head;
        size_t num = 0;
        while (cur->next) {
            num++;
            cur = cur->next;
        }
    }
    void print() {
        node* cur = _head->next;
        while (cur) {
            cout << cur->val << " ";
            cur = cur->next;
        }
        cout << endl;
    }
private:
```

```
    node* _head;//哨兵
};
```

问题七

1.代码中使用了`Book b2(b1)`却没有拷贝构造

2.set_name使用了`const`修饰符，不能在函数中修改`name`

3.没有初始化`BookCnt`

修改后为：

```
#include <iostream>
#include <cstring>
using namespace std;
class Book {
    static int BookCnt;
    char* name;
public:
    Book(const char* _name);
    Book(const Book& book);
    ~Book();
    char* get_name() const;
    void set_name(const char* _name);
};
Book::Book(const char* _name) {
    name = new char[strlen(_name) + 1];
    strcpy(name, _name);
    BookCnt++;
}
Book::Book(const Book& book){
    delete[]name;
    name = new char[strlen(book.name) + 1];
    strcpy(name, book.name);
    BookCnt++;
}
Book::~Book() {
    delete[]name;
    name = nullptr;
}
char* Book::get_name() const {
    return name;
}
void Book::set_name(const char* _name){
    delete[]name;
    name = new char[strlen(_name) + 1];
    strcpy(name, _name);
}
int Book::BookCnt = 0;
int main() {
    Book b1("Computer␣Science");
    Book b2(b1);
    return 0;
}
```