1. Object-c的类可以多重继承么?可以实现多个接口么?Category是什么?重写一个类的方式用继承好还是分类好?为什么?
答： Object-c的类不可以多重继承;可以实现多个接口，通过实现多个接口可以完成C++的多重继承;Category是类别，一般情况用分类好，用Category去重写类的方法，仅对本Category有效，不会影响到其他类与原有类的关系。
2. #import 跟#include 又什么区别，@class呢, #import<> 跟 #import""又什么区别?
答：#import是Objective-C导入头文件的关键字，#include是C/C++导入头文件的关键字,使用#import头文件会自动只导入一次，不会重复导入，相当于#include和#pragma once;@class告诉编译器某个类的声明，当执行时，才去查看类的实现文件，可以解决头文件的相互包含;#import<>用来包含系统的头文件，#import""用来包含用户头文件。
3. 属性readwrite，readonly，assign，retain，copy，nonatomic 各是什么作用，在那种情况下用?
答：
1). readwrite 是可读可写特性;需要生成getter方法和setter方法时
2). readonly 是只读特性 只会生成getter方法 不会生成setter方法 ;不希望属性在类外改变
3). assign 是赋值特性，setter方法将传入参数赋值给实例变量;仅设置变量时;
4). retain 表示持有特性，setter方法将传入参数先保留，再赋值，传入参数的retaincount会+1;
5). copy 表示赋值特性，setter方法将传入对象复制一份;需要完全一份新的变量时。
6).nonatomic 非原子操作，决定编译器生成的setter getter是否是原子操作，atomic表示多线程安全，一般使用nonatomic
4.写一个setter方法用于完成@property (nonatomic,retain)NSString *name,写一个setter方法用于完成@property(nonatomic，copy)NSString *name
答：

```
-?(void)?setName:(NSString*)?str
{
[str?retain];
[name?release];
name?=?str;
}
-?(void)setName:(NSString?*)str
{
id?t?=?[str?copy];
[name?release];
name?=?t;
}
```

5.对于语句NSString*obj = [[NSData alloc] init]; obj在编译时和运行时分别时什么类型的对象?
答： 编译时是NSString的类型;运行时是NSData类型的对象
6.常见的object-c的数据类型有那些， 和C的基本数据类型有什么区别?如：NSInteger和int
答：object-c的数据类型有NSString，NSNumber，NSArray，NSMutableArray，NSData等等，这些都是class，创建后便是对象，而C语言的基本数据类型int，只是一定字节的内存空间，用于存放数值;NSInteger是基本数据类型，并不是NSNumber的子类，当然也不是NSObject的子类。
NSInteger是基本数据类型Int或者Long的别名(NSInteger的定义typedef long NSInteger)，它的区别在于，NSInteger会根据系统是32位还是64位来决定是本身是int还是Long。
7.id 声明的对象有什么特性?
答：Id 声明的对象具有运行时的特性，即可以指向任意类型的objcetive-c的对象;
8.Objective-C如何对内存管理的,说说你的看法和解决方法?
答：Objective-C的内存管理主要有三种方式ARC(自动内存计数)、手动内存计数、内存池。

1). (Garbage Collection)自动内存计数：这种方式和Java类似，在你的程序的执行过程中。始终有一个高人在背后准确地帮你收拾垃圾，你不用考虑它什么时候开始工作，怎样工作。你只需要明白，我申请了一段内存空间，当我不再使用从而这段内存成为垃圾的时候，我就彻底的把它忘记掉，反正那个高人会帮我收拾垃圾。遗憾的是，那个高人需要消耗一定的资源，在携带设备里面，资源是紧俏商品所以iPhone不支持这个功能。所以“Garbage Collection”不是本入门指南的范围，对“Garbage Collection”内部机制感兴趣的同学可以参考一些其他的资料，不过说老实话“Garbage Collection”不大适合适初学者研究。
解决: 通过alloc – initial方式创建的, 创建后引用计数+1, 此后每retain一次引用计数+1, 那么在程序中做相应次数的release就好了.
2). (Reference Counted)手动内存计数：就是说，从一段内存被申请之后，就存在一个变量用于保存这段内存被使用的次数，我们暂时把它称为计数器，当计数器变为0的时候，那么就是释放这段内存的时候。比如说，当在程序A里面一段内存被成功申请完成之后，那么这个计数器就从0变成1(我们把这个过程叫做alloc)，然后程序B也需要使用这个内存，那么计数器就从1变成了2(我们把这个过程叫做retain)。紧接着程序A不再需要这段内存了，那么程序A就把这个计数器减1(我们把这个过程叫做release);程序B也不再需要这段内存的时候，那么也把计数器减1(这个过程还是release)。当系统(也就是Foundation)发现这个计数器变 成员了0，那么就会调用内存回收程序把这段内存回收(我们把这个过程叫做dealloc)。顺便提一句，如果没有Foundation，那么维护计数器，释放内存等等工作需要你手工来完成。
解决:一般是由类的静态方法创建的, 函数名中不会出现alloc或init字样, 如[NSString string]和[NSArray arrayWithObject:], 创建后引用计数+0, 在函数出栈后释放, 即相当于一个栈上的局部变量. 当然也可以通过retain延长对象的生存期.
3). (NSAutoRealeasePool)内存池：可以通过创建和释放内存池控制内存申请和回收的时机.
解决:是由autorelease加入系统内存池, 内存池是可以嵌套的, 每个内存池都需要有一个创建释放对, 就像main函数中写的一样. 使用也很简单, 比如[[[NSString alloc]initialWithFormat:@”Hey you!”] autorelease], 即将一个NSString对象加入到最内层的系统内存池, 当我们释放这个内存池时, 其中的对象都会被释放.
9. 原子(atomic)跟非原子(non-atomic)属性有什么区别?
答：
1). atomic提供多线程安全。是防止在写未完成的时候被另外一个线程读取，造成数据错误
2). non-atomic:在自己管理内存的环境中，解析的访问器保留并自动释放返回的值，如果指定了nonatomic ，那么访问器只是简单地返回这个值。
10. 看下面的程序,第一个NSLog会输出什么?这时str的retainCount是多少?第二个和第三个呢? 为什么?

```
NSMutableArray*?ary?=?[[NSMutableArray?array]?retain];
NSString?*str?=?[NSString?stringWithFormat:@"test"];
[str?retain];
[aryaddObject:str];
NSLog(@”%@%d”,str,[str?retainCount]);
[str?retain];
[str?release];
[str?release];
NSLog(@”%@%d”,str,[str?retainCount]);
[aryremoveAllObjects];
NSLog(@”%@%d”,str,[str?retainCount]);
```

str的retainCount创建+1，retain+1，加入数组自动+1 3
retain+1，release-1，release-1 2

数组删除所有对象，所有数组内的对象自动-1 1
11. 内存管理的几条原则时什么?按照默认法则.那些关键字生成的对象需要手动释放?在和property结合的时候怎样有效的避免内存泄露?
答：谁申请，谁释放
遵循Cocoa Touch的使用原则;
内存管理主要要避免“过早释放”和“内存泄漏”，对于“过早释放”需要注意@property设置特性时，一定要用对特性关键字，对于“内存泄漏”，一定要申请了要负责释放，要细心。
关键字alloc 或new 生成的对象需要手动释放;
设置正确的property属性，对于retain需要在合适的地方释放，
12.如何对iOS设备进行性能测试?
答： Profile-> Instruments ->Time Profiler
13. Object C中创建线程的方法是什么?如果在主线程中执行代码，方法是什么?如果想延时执行代码、方法又是什么?
答：线程创建有三种方法：使用NSThread创建、使用GCD的dispatch、使用子类化的NSOperation,然后将其加入NSOperationQueue;在主线程执行代码，方法是performSelectorOnMainThread，如果想延时执行代码可以用performSelector:onThread:withObject:waitUntilDone:
14. MVC设计模式是什么？ 你还熟悉什么设计模式?
答:
设计模式：并不是一种新技术，而是一种编码经验，使用比如java中的接口，iphone中的协议，继承关系等基本手段，用比较成熟的逻辑去处理某一种类型的事情，总结为所谓设计模式。面向对象编程中，java已经归纳了23种设计模式。
mvc设计模式 ： 模型，视图，控制器，可以将整个应用程序在思想上分成三大块，对应是的数据的存储或处理，前台的显示，业务逻辑的控制。 Iphone本身的设计思想就是遵循mvc设计模式。其不属于23种设计模式范畴。
代理模式：代理模式给某一个对象提供一个代理对象，并由代理对象控制对源对象的引用.比如一个工厂生产了产品，并不想直接卖给用户，而是搞了很多代理商，用户可以直接找代理商买东西，代理商从工厂进货.常见的如QQ的自动回复就属于代理拦截，代理模式在iphone中得到广泛应用.
单例模式：说白了就是一个类不通过alloc方式创建对象，而是用一个静态方法返回这个类的对象。系统只需要拥有一个的全局对象，这样有利于我们协调系统整体的行为，比如想获得[UIApplication sharedApplication];任何地方调用都可以得到 UIApplication的对象，这个对象是全局唯一的。
观察者模式： 当一个物体发生变化时，会通知所有观察这个物体的观察者让其做出反应。实现起来无非就是把所有观察者的对象给这个物体，当这个物体的发生改变，就会调用遍历所有观察者的对象调用观察者的方法从而达到通知观察者的目的。
工厂模式:

```
public?class?Factory{
public?static?Sample?creator(int?which){
if?(which==1)
return?new?SampleA();
else?if?(which==2)
return?new?SampleB();
}
}
```

15 浅复制和深复制的区别?
答：浅层复制：只复制指向对象的指针，而不复制引用对象本身。
深层复制：复制引用对象本身。

意思就是说我有个A对象，复制一份后得到A_copy对象后，对于浅复制来说，A和A_copy指向的是同一个内存资源，复制的只不过是是一个指针，对象本身资源
还是只有一份，那如果我们对A_copy执行了修改操作,那么发现A引用的对象同样被修改，这其实违背了我们复制拷贝的一个思想。深复制就好理解了,内存中存在了
两份独立对象本身。
用网上一哥们通俗的话将就是：
浅复制好比你和你的影子，你完蛋，你的影子也完蛋
深复制好比你和你的克隆人，你完蛋，你的克隆人还活着。
16. 类别的作用?继承和类别在实现中有何区别?
答：category 可以在不获悉，不改变原来代码的情况下往里面添加新的方法，只能添加，不能删除修改，并且如果类别和原来类中的方法产生名称冲突，则类别将覆盖原来的方法，因为类别具有更高的优先级。
类别主要有3个作用：
1).将类的实现分散到多个不同文件或多个不同框架中。
2).创建对私有方法的前向引用。
3).向对象添加非正式协议。
继承可以增加，修改或者删除方法，并且可以增加属性。
17. 类别和类扩展的区别。
答：category和extensions的不同在于 后者可以添加属性。另外后者添加的方法是必须要实现的。extensions可以认为是一个私有的Category。
18. oc中的协议和java中的接口概念有何不同?
答：OC中的代理有2层含义，官方定义为 formal和informal protocol。前者和Java接口一样。
informal protocol中的方法属于设计模式考虑范畴，不是必须实现的，但是如果有实现，就会改变类的属性。
其实关于正式协议，类别和非正式协议我很早前学习的时候大致看过，也写在了学习教程里
“非正式协议概念其实就是类别的另一种表达方式“这里有一些你可能希望实现的方法，你可以使用他们更好的完成工作”。
这个意思是，这些是可选的。比如我门要一个更好的方法，我们就会申明一个这样的类别去实现。然后你在后期可以直接使用这些更好的方法。
这么看，总觉得类别这玩意儿有点像协议的可选协议。”
现在来看，其实protocal已经开始对两者都统一和规范起来操作，因为资料中说“非正式协议使用interface修饰“，
现在我们看到协议中两个修饰词：“必须实现(@requied)”和“可选实现(@optional)”。
19. 什么是KVO和KVC?
答：KVC:键 – 值编码是一种间接访问对象的属性使用字符串来标识属性，而不是通过调用存取方法，直接或通过实例变量访问的机制。
很多情况下可以简化程序代码。apple文档其实给了一个很好的例子。
KVO:键值观察机制，他提供了观察某一属性变化的方法，极大的简化了代码。
具体用看到嗯哼用到过的一个地方是对于按钮点击变化状态的的监控。
比如我自定义的一个button

```
[self?addObserver:self?forKeyPath:@"highlighted"?options:0?context:nil];
#pragma?mark?KVO
-?(void)observeValueForKeyPath:(NSString?*)keyPath?ofObject:(id)object?change:(NSDictionary?*)change?context:(void?*)context
{
```

```
if?([keyPath?isEqualToString:@"highlighted"]?)?{
[self?setNeedsDisplay];
}
}
```
对于系统是根据keypath去取的到相应的值发生改变，理论上来说是和kvc机制的道理是一样的。
对于kvc机制如何通过key寻找到value：
“当通过KVC调用对象时，比如：[self valueForKey:@”someKey”]时，程序会自动试图通过几种不同的方式解析这个调用。首先查找对象是否带有 someKey 这个方法，如果没找到，会继续查找对象是否带有someKey这个实例变量(iVar)，如果还没有找到，程序会继续试图调用 -(id) valueForUndefinedKey:这个方法。如果这个方法还是没有被实现的话，程序会抛出一个NSUndefinedKeyException异常错误。
(cocoachina.com注：Key-Value Coding查找方法的时候，不仅仅会查找someKey这个方法，还会查找getsomeKey这个方法，前面加一个get，或者_someKey以及_getsomeKey这几种形式。同时，查找实例变量的时候也会不仅仅查找someKey这个变量，也会查找_someKey这个变量是否存在。)
设计valueForUndefinedKey:方法的主要目的是当你使用-(id)valueForKey方法从对象中请求值时，对象能够在错误发生前，有最后的机会响应这个请求。这样做有很多好处，下面的两个例子说明了这样做的好处。“
来至cocoa，这个说法应该挺有道理。
因为我们知道button却是存在一个highlighted实例变量.因此为何上面我们只是add一个相关的keypath就行了，
可以按照kvc查找的逻辑理解，就说的过去了。
20. 代理的作用?
答：代理的目的是改变或传递控制链。允许一个类在某些特定时刻通知到其他类，而不需要获取到那些类的指针。可以减少框架复杂度。
另外一点，代理可以理解为java中的回调监听机制的一种类似。
21. oc中可修改和不可以修改类型。
答：可修改不可修改的集合类。这个我个人简单理解就是可动态添加修改和不可动态添加修改一样。

比如NSArray和NSMutableArray。前者在初始化后的内存控件就是固定不可变的，后者可以添加等，可以动态申请新的内存空间。
22. 我们说的oc是动态运行时语言是什么意思?
答：多态。 主要是将数据类型的确定由编译时，推迟到了运行时。
这个问题其实浅涉及到两个概念，运行时和多态。
简单来说，运行时机制使我们直到运行时才去决定一个对象的类别，以及调用该类别对象指定方法。

多态：不同对象以自己的方式响应相同的消息的能力叫做多态。意思就是假设生物类(life)都用有一个相同的方法-eat;
那人类属于生物，猪也属于生物，都继承了life后，实现各自的eat，但是调用是我们只需调用各自的eat方法。
也就是不同的对象以自己的方式响应了相同的消息(响应了eat这个选择器)。
因此也可以说，运行时机制是多态的基础?~~~
23. 通知和协议的不同之处?
答：协议有控制链(has-a)的关系，通知没有。
首先我一开始也不太明白，什么叫控制链(专业术语了~)。但是简单分析下通知和代理的行为模式，我们大致可以有自己的理解
简单来说，通知的话，它可以一对多，一条消息可以发送给多个消息接受者。

代理按我们的理解，到不是直接说不能一对多，比如我们知道的明星经济代理人，很多时候一个经济人负责好几个明星的事务。
只是对于不同明星间，代理的事物对象都是不一样的，一一对应，不可能说明天要处理A明星要一个发布会，代理人发出处理发布会的消息后，别称B的
发布会了。但是通知就不一样，他只关心发出通知，而不关心多少接收到感兴趣要处理。
因此控制链(has-a从英语单词大致可以看出，单一拥有和可控制的对应关系。
24. 什么是推送消息?
答：推送通知更是一种技术。
简单点就是客户端获取资源的一种手段。
普通情况下，都是客户端主动的pull。
推送则是服务器端主动push。 测试push的实现可以查看该博文。
25. 关于多态性
答：多态，子类指针可以赋值给父类。
这个题目其实可以出到一切面向对象语言中，
因此关于多态，继承和封装基本最好都有个自我意识的理解，也并非一定要把书上资料上写的能背出来
26. 对于单例的理解
答：在objective-c中要实现一个单例类，至少需要做以下四个步骤：
1).为单例对象实现一个静态实例，并初始化，然后设置成nil，
2).实现一个实例构造方法检查上面声明的静态实例是否为nil，如果是则新建并返回一个本类的实例，
3).重写allocWithZone方法，用来保证其他人直接使用alloc和init试图获得一个新实力的时候不产生一个新实例，
4).适当实现allocWitheZone，copyWithZone，release和autorelease。
27. 说说响应链
答： 事件响应链。包括点击事件，画面刷新事件等。在视图栈内从上至下，或者从下之上传播。
可以说点事件的分发，传递以及处理。具体可以去看下touch事件这块。因为问的太抽象化了
严重怀疑题目出到越后面就越笼统。
可以从责任链模式，来讲通过事件响应链处理，其拥有的扩展性
28. frame和bounds有什么不同?
答:frame指的是：该view在父view坐标系统中的位置和大小。(参照点是父亲的坐标系统)
bounds指的是：该view在本身坐标系统中 的位置和大小。(参照点是本身坐标系统)
29. 方法和选择器有何不同?
答：selector是一个方法的名字，method是一个组合体，包含了名字和实现.
详情可以看apple文档。
30. OC的垃圾回收机制?
答： OC2.0有Garbage collection，但是iOS平台不提供。
一般我们了解的objective-c对于内存管理都是手动操作的，但是也有自动释放池。
但是差了大部分资料，貌似不要和arc机制搞混就好了。
31. NSOperation queue?
答：存放NSOperation的集合类。
操作和操作队列，基本可以看成java中的线程和线程池的概念。用于处理ios多线程开发的问题。
网上部分资料提到一点是，虽然是queue，但是却并不是带有队列的概念，放入的操作并非是按照严格的先进现出。

这边又有个疑点是，对于队列来说，先进先出的概念是Afunc添加进队列，Bfunc紧跟着也进入队列，Afunc先执行这个是必然的，
但是Bfunc是等Afunc完全操作完以后，B才开始启动并且执行，因此队列的概念离乱上有点违背了多线程处理这个概念。
但是转念一想其实可以参考银行的取票和叫号系统。
因此对于A比B先排队取票但是B率先执行完操作，我们亦然可以感性认为这还是一个队列。
但是后来看到一票关于这操作队列话题的文章，其中有一句提到
“因为两个操作提交的时间间隔很近，线程池中的线程，谁先启动是不定的。”
瞬间觉得这个queue名字有点忽悠人了，还不如pool~
综合一点，我们知道他可以比较大的用处在于可以帮组多线程编程就好了。
32. 什么是延迟加载?
答：懒汉模式，只在用到的时候才去初始化。
也可以理解成延时加载。
我觉得最好也最简单的一个列子就是tableView中图片的加载显示了。
一个延时载，避免内存过高，一个异步加载，避免线程堵塞。
33. 是否在一个视图控制器中嵌入两个tableview控制器?
答：一个视图控制只提供了一个View视图，理论上一个tableViewController也不能放吧，
只能说可以嵌入一个tableview视图。当然，题目本身也有歧义，如果不是我们定性思维认为的UIViewController，而是宏观的表示视图控制者，那我们倒是可以把其看成一个视图控制者，它可以控制多个视图控制器，比如TabbarController那样的感觉。
34. 一个tableView是否可以关联两个不同的数据源?你会怎么处理?
答：首先我们从代码来看，数据源如何关联上的，其实是在数据源关联的代理方法里实现的。
因此我们并不关心如何去关联他，他怎么关联上，方法只是让我返回根据自己的需要去设置如相关的数据源。
因此，我觉得可以设置多个数据源啊，但是有个问题是，你这是想干嘛呢?想让列表如何显示，不同的数据源分区块显示?
35. 什么时候使用NSMutableArray，什么时候使用NSArray?
答：当数组在程序运行时，需要不断变化的，使用NSMutableArray，当数组在初始化后，便不再改变的，使用NSArray。需要指出的是，使用NSArray只表明的是该数组在运行时不发生改变，即不能往NSAarry的数组里新增和删除元素，但不表明其数组内的元素的内容不能发生改变。NSArray是线程安全的，NSMutableArray不是线程安全的，多线程使用到NSMutableArray需要注意。
36. 给出委托方法的实例，并且说出UITableVIew的Data Source方法
答：CocoaTouch框架中用到了大量委托，其中UITableViewDelegate就是委托机制的典型应用，是一个典型的使用委托来实现适配器模式，其中UITableViewDelegate协议是目标，tableview是适配器，实现UITableViewDelegate协议，并将自身设置为talbeview的delegate的对象，是被适配器，一般情况下该对象是UITableViewController。
UITableVIew的Data Source方法有- (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section;
- (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath *)indexPath;
37. 在应用中可以创建多少autorelease对象，是否有限制?
答案：无
38. 如果我们不创建内存池，是否有内存池提供给我们?
答:界面线程维护着自己的内存池，用户自己创建的数据线程，则需要创建该线程的内存池
39. 什么时候需要在程序中创建内存池?
答：用户自己创建的数据线程，则需要创建该线程的内存池

40. 类NSObject的那些方法经常被使用?
答：NSObject是Objetive-C的基类，其由NSObject类及一系列协议构成。
其中类方法alloc、class、 description 对象方法init、dealloc、–performSelector:withObject:afterDelay:等经常被使用
41. 什么是简便构造方法?
答：简便构造方法一般由CocoaTouch框架提供，如NSNumber的 + numberWithBool: + numberWithChar: + numberWithDouble: + numberWithFloat: + numberWithInt:
Foundation下大部分类均有简便构造方法，我们可以通过简便构造方法，获得系统给我们创建好的对象，并且不需要手动释放。
42. 如何使用Xcode设计通用应用?
答：使用MVC模式设计应用，其中Model层完成脱离界面，即在Model层，其是可运行在任何设备上，在controller层，根据iPhone与iPad(独有UISplitViewController)的不同特点选择不同的viewController对象。在View层，可根据现实要求，来设计，其中以xib文件设计时，其设置其为universal。
43. UIView的动画效果有那些?
答：有很多，如 UIViewAnimationOptionCurveEaseInOut UIViewAnimationOptionCurveEaseIn UIViewAnimationOptionCurveEaseOut UIViewAnimationOptionTransitionFlipFromLeft UIViewAnimationOptionTransitionFlipFromRight UIViewAnimationOptionTransitionCurlUpUIViewAnimationOptionTransitionCurlDown
44. 在iPhone应用中如何保存数据?
答：有以下几种保存机制：
1).通过web服务，保存在服务器上
2).通过NSCoder固化机制，将对象保存在文件中
3).通过SQlite或CoreData保存在文件数据库中
45. 什么是coredata?
答：coredata是苹果提供一套数据保存框架，其基于SQlite
46. 什么是NSManagedObject模型?
答：NSManagedObject是NSObject的子类 ，也是coredata的重要组成部分，它是一个通用的类,实现了core data 模型层所需的基本功能，用户可通过子类化NSManagedObject，建立自己的数据模型。
47. 什么是NSManagedobjectContext?
答：NSManagedobjectContext对象负责应用和数据库之间的交互。
48. 什么是谓词?
答：谓词是通过NSPredicate，是通过给定的逻辑条件作为约束条件，完成对数据的筛选。
1
2
```
predicate?=?[NSPredicate?predicateWithFormat:@"customerID?==?%d",n];
a?=?[customers?filteredArrayUsingPredicate:predicate];
```
49. 和coredata一起有哪几种持久化存储机制?
答：存入到文件、 存入到NSUserDefaults(系统plist文件中)、存入到Sqlite文件数据库
50. 谈谈对Block 的理解?并写出一个使用Block执行UIVew动画?
答：Block是可以获取其他函数局部变量的匿名函数，其不但方便开发，并且可以大幅提高应用的执行效率(多核心CPU可直接处理Block指令)
1
2
3
4

5
[UIView?transitionWithView:self.view
duration:0.2
options:UIViewAnimationOptionTransitionFlipFromLeft
animations:^{?[[blueViewController?view]?removeFromSuperview];?[[self?view]?
insertSubview:yellowViewController.view?atIndex:0];?}
completion:NULL];
51. 写出上面代码的Block的定义。
答:
1
2
typedef?void(^animations)?(void);
typedef?void(^completion)?(BOOL?finished);
52. 试着使用+ beginAnimations:context:以及上述Block的定义，写出一个可以完成
1
+?(void)transitionWithView:(UIView?*)view?duration:(NSTimeInterval)duration?options:
(UIViewAnimationOptions)options?animations:(void?(^)(void))animations?completion:(void?(^)
(BOOL?finished))completion?NS_AVAILABLE_IOS(4_0);
操作的函数执行部分
答案：无
53. 做过的项目是否涉及网络访问功能，使用什么对象完成网络功能?
答：ASIHTTPRequest与NSURLConnection
54. 简单介绍下NSURLConnection类及+ sendSynchronousRequest:returningResponse:error:与– initWithRequest:delegate:两个方法的区别?
答: NSURLConnection主要用于网络访问，其中+ sendSynchronousRequest:returningResponse:error:是同步访问数据，即当前线程会阻塞，并等待request的返回的response，而– initWithRequest:delegate:使用的是异步加载，当其完成网络访问后，会通过delegate回到主线程，并其委托的对象。
55. 多线程是什么
答: 多线程是个复杂的概念，按字面意思是同步完成多项任务，提高了资源的使用效率，从硬件、操作系统、应用软件不同的角度去看，多线程被赋予不同的内涵，对于硬件，现在市面上多数的CPU都是多核的，多核的CPU运算多线程更为出色;从操作系统角度，是多任务，现在用的主流操作系统都是多任务的，可以一边听歌、一边写博客;对于应用来说，多线程可以让应用有更快的回应，可以在网络下载时，同时响应用户的触摸操作。在iOS应用中，对多线程最初的理解，就是并发，它的含义是原来先做烧水，再摘菜，再炒菜的工作，会变成烧水的同时去摘菜，最后去炒菜。
56. iOS 中的多线程
答: iOS中的多线程，是Cocoa框架下的多线程，通过Cocoa的封装，可以让我们更为方便的使用线程，做过C++的同学可能会对线程有更多的理解，比如线程的创立，信号量、共享变量有认识，Cocoa框架下会方便很多，它对线程做了封装，有些封装，可以让我们创建的对象，本身便拥有线程，也就是线程的对象化抽象，从而减少我们的工程，提供程序的健壮性。
GCD是(Grand Central Dispatch)的缩写 ，从系统级别提供的一个易用地多线程类库，具有运行时的特点，能充分利用多核心硬件。GCD的API接口为C语言的函数，函数参数中多数有Block，关于Block的使用参看这里，为我们提供强大的“接口”，对于GCD的使用参见本文
NSOperation与Queue
NSOperation是一个抽象类，它封装了线程的细节实现，我们可以通过子类化该对象，加上NSQueue来同面向对象的思维，管理多线程程序。具体可参看这里：一个基于NSOperation的多线程网络访问的项目。
NSThread

NSThread是一个控制线程执行的对象，它不如NSOperation抽象，通过它我们可以方便的得到一个线程，并控制它。但NSThread的线程之间的并发控制，是需要我们自己来控制的，可以通过NSCondition实现。
参看 iOS多线程编程之NSThread的使用
其他多线程
在Cocoa的框架下，通知、Timer和异步函数等都有使用多线程，(待补充).
57. 在项目什么时候选择使用GCD，什么时候选择NSOperation?
答: 项目中使用NSOperation的优点是NSOperation是对线程的高度抽象，在项目中使用它，会使项目的程序结构更好，子类化NSOperation的设计思路，是具有面向对象的优点(复用、封装)，使得实现是多线程支持，而接口简单，建议在复杂项目中使用。
项目中使用GCD的优点是GCD本身非常简单、易用，对于不复杂的多线程操作，会节省代码量，而Block参数的使用，会是代码更为易读，建议在简单项目中使用。
58. 什么是block
答: 对于闭包(block),有很多定义，其中闭包就是能够读取其它函数内部变量的函数，这个定义即接近本质又较好理解。对于刚接触Block的同学，会觉得有些绕，因为我们习惯写这样的程序main(){ funA();} funA(){funB();} funB(){…..}; 就是函数main调用函数A，函数A调用函数B… 函数们依次顺序执行，但现实中不全是这样的，例如项目经理M，手下有3个程序员A、B、C，当他给程序员A安排实现功能F1时，他并不等着A完成之后，再去安排B去实现F2，而是安排给A功能F1，B功能F2，C功能F3，然后可能去写技术文档，而当A遇到问题时，他会来找项目经理M，当B做完时，会通知M，这就是一个异步执行的例子。在这种情形下，Block便可大显身手，因为在项目经理M，给A安排工作时，同时会告诉A若果遇到困难，如何能找到他报告问题(例如打他手机号)，这就是项目经理M给A的一个回调接口，要回掉的操作，比如接到电话，百度查询后，返回网页内容给A，这就是一个Block，在M交待工作时，已经定义好，并且取得了F1的任务号(局部变量)，却是在当A遇到问题时，才调用执行，跨函数在项目经理M查询百度，获得结果后回调该block。
59. block 实现原理
答: Objective-C是对C语言的扩展，block的实现是基于指针和函数指针。
从计算语言的发展，最早的goto，高级语言的指针，到面向对象语言的block，从机器的思维，一步步接近人的思维，以方便开发人员更为高效、直接的描述出现实的逻辑(需求)。
使用实例
cocoaTouch框架下动画效果的Block的调用
使用typed声明block
1
2
typedef?void(^didFinishBlock)?(NSObject?*ob);
这就声明了一个didFinishBlock类型的block,
然后便可用
1
@property?(nonatomic,copy)?didFinishBlock?finishBlock;
声明一个blokc对象，注意对象属性设置为copy，接到block 参数时，便会自动复制一份。
__block是一种特殊类型，
使用该关键字声明的局部变量，可以被block所改变，并且其在原函数中的值会被改变。
60.关于block
答: 面试时，面试官会先问一些，是否了解block，是否使用过block，这些问题相当于开场白，往往是下面一系列问题的开始，所以一定要如实根据自己的情况回答。
1). 使用block和使用delegate完成委托模式有什么优点?

首先要了解什么是委托模式，委托模式在iOS中大量应用，其在设计模式中是适配器模式中的对象适配器，Objective-C中使用id类型指向一切对象，使委托模式更为简洁。了解委托模式的细节：
iOS设计模式一-委托模式
使用block实现委托模式，其优点是回调的block代码块定义在委托对象函数内部，使代码更为紧凑;适配对象不再需要实现具体某个protocol，代码更为简洁。
2). 多线程与block
GCD与Block
使用 dispatch_async 系列方法，可以以指定的方式执行block
GCD编程实例
dispatch_async的完整定义

```
void?dispatch_async(
dispatch_queue_t?queue,
dispatch_block_t?block);
```

功能：在指定的队列里提交一个异步执行的block，不阻塞当前线程
通过queue来控制block执行的线程。主线程执行前文定义的 finishBlock对象

```
1
dispatch_async(dispatch_get_main_queue(),^(void){finishBlock();});
```

62.谈谈Object-C的内存管理方式及过程?
答: 1).当你使用new,alloc和copy方法创建一个对象时,该对象的保留计数器值为1.当你不再使用该对象时,你要负责向该对象发送一条release或autorelease消息.这样,该对象将在使用寿命结束时被销毁.
2).当你通过任何其他方法获得一个对象时,则假设该对象的保留计数器值为1,而且已经被设置为自动释放,你不需要执行任何操作来确保该对象被清理.如果你打算在一段时间内拥有该对象,则需要保留它并确保在操作完成时释放它.
3).如果你保留了某个对象,你需要(最终)释放或自动释放该对象.必须保持retain方法和release方法的使用次数相等.
63.Object-C有私有方法吗? 私有变量呢?
答: objective-c – 类里面的方法只有两种, 静态方法和实例方法. 这似乎就不是完整的面向对象了,按照OO的原则就是一个对象只暴露有用的东西. 如果没有了私有方法的话, 对于一些小范围的代码重用就不那么顺手了. 在类里面声名一个私有方法

```
@interface?Controller?:?NSObject?{?NSString?*something;?}
+?(void)thisIsAStaticMethod;
–?(void)thisIsAnInstanceMethod;
@end
@interface?Controller?(private)?-
(void)thisIsAPrivateMethod;
@end
```

@private可以用来修饰私有变量
在Objective-C中，所有实例变量默认都是私有的，所有实例方法默认都是公有的
64.Object-C有多继承吗? 没有的话用什么代替? cocoa 中所有的类都是NSObject 的子类
答: 多继承在这里是用protocol 委托代理 来实现的
你不用去考虑繁琐的多继承 ,虚基类的概念.
ood的多态特性 在 obj-c 中通过委托来实现.
65.内存管理 Autorelease、retain、copy、assign的set方法和含义?
答: 1).你初始化(alloc/init)的对象，你需要释放(release)它。例如：
NSMutableArray aArray = [[NSArray alloc] init]; 后，需要 [aArray release];
2).你retain或copy的，你需要释放它。例如：

[aArray retain] 后，需要 [aArray release];
3).被传递(assign)的对象，你需要斟酌的retain和release。例如：
obj2 = [[obj1 someMethod] autorelease];
对象2接收对象1的一个自动释放的值，或传递一个基本数据类型(NSInteger，NSString)时：你或希望将对象2进行retain，以防止它在被使用之前就被自动释放掉。但是在retain后，一定要在适当的时候进行释放。
关于索引计数(Reference Counting)的问题
retain值 = 索引计数(Reference Counting)
NSArray对象会retain(retain值加一)任何数组中的对象。当NSArray被卸载(dealloc)的时候，所有数组中的对象会 被 执行一次释放(retain值减一)。不仅仅是NSArray，任何收集类(Collection Classes)都执行类似操作。例如 NSDictionary，甚至UINavigationController。
Alloc/init建立的对象，索引计数为1。无需将其再次retain。
[NSArray array]和[NSDate date]等“方法”建立一个索引计数为1的对象，但是也是一个自动释放对象。所以是本地临时对象，那么无所谓了。如果是打算在全Class中使用的变量(iVar)，则必须retain它。
缺省的类方法返回值都被执行了“自动释放”方法。(*如上中的NSArray)
在类中的卸载方法“dealloc”中，release所有未被平衡的NS对象。(*所有未被autorelease，而retain值为1的)
66. C和obj-c 如何混用
答: 1).obj-c的编译器处理后缀为m的文件时，可以识别obj-c和c的代码，处理mm文件可以识别obj-c,c,c++代码，但cpp文件必须只能用c/c++代码，而且cpp文件include的头文件中，也不能出现obj-c的代码，因为cpp只是cpp
2).在mm文件中混用cpp直接使用即可，所以obj-c混cpp不是问题
3).在cpp中混用obj-c其实就是使用obj-c编写的模块是我们想要的。
如果模块以类实现，那么要按照cpp class的标准写类的定义，头文件中不能出现obj-c的东西，包括#import cocoa的。实现文件中，即类的实现代码中可以使用obj-c的东西，可以import,只是后缀是mm。
如果模块以函数实现，那么头文件要按c的格式声明函数，实现文件中，c++函数内部可以用obj-c，但后缀还是mm或m。
总结：只要cpp文件和cpp include的文件中不包含obj-c的东西就可以用了，cpp混用obj-c的关键是使用接口，而不能直接使用 实现代 码，实际上cpp混用的是obj-c编译后的o文件，这个东西其实是无差别的，所以可以用。obj-c的编译器支持cpp
67. Objective-C堆和栈的区别?
答: 管理方式：对于栈来讲，是由编译器自动管理，无需我们手工控制；对于堆来说，释放工作由程序员控制，容易产生memory leak。
申请大小：
栈：在Windows下,栈是向低地址扩展的数据结构，是一块连续的内存的区域。这句话的意思是栈顶的地址和栈的最大容量是系统预先规定好的，在 WINDOWS下，栈的大小是2M（也有的说是1M，总之是一个编译时就确定的常数），如果申请的空间超过栈的剩余空间时，将提示overflow。因此，能从栈获得的空间较小。
堆：堆是向高地址扩展的数据结构，是不连续的内存区域。这是由于系统是用链表来存储的空闲内存地址的，自然是不连续的，而链表的遍历方向是由低地址向高地址。堆的大小受限于计算机系统中有效的虚拟内存。由此可见，堆获得的空间比较灵活，也比较大。

碎片问题：对于堆来讲，频繁的new/delete势必会造成内存空间的不连续，从而造成大量的碎片，使程序效率降低。对于栈来讲，则不会存在这个问题，因为栈是先进后出的队列，他们是如此的一一对应，以至于永远都不可能有一个内存块从栈中间弹出
分配方式：堆都是动态分配的，没有静态分配的堆。栈有2种分配方式：静态分配和动态分配。静态分配是编译器完成的，比如局部变量的分配。动态分配由alloca函数进行分配，但是栈的动态分配和堆是不同的，他的动态分配是由编译器进行释放，无需我们手工实现。
分配效率：栈是机器系统提供的数据结构，计算机会在底层对栈提供支持：分配专门的寄存器存放栈的地址，压栈出栈都有专门的指令执行，这就决定了栈的效率比较高。堆则是C/C++函数库提供的，它的机制是很复杂的。
68. ViewController的didReceiveMemoryWarning怎么被调用：
答:[supper didReceiveMemoryWarning];
69.什么时候用delegate,什么时候用Notification?
答: delegate针对one-to-one关系，用于sender接受到reciever的某个功能反馈值。
notification针对one-to-one/many/none,reciver,用于通知多个object某个事件。
70.用预处理指令#define声明一个常数，用以表明1年中有多少秒（忽略闰年问题）
答:
#define SECONDS_PER_YEAR (60 * 60 * 24 * 365)UL
我在这想看到几件事情：
#define 语法的基本知识（例如：不能以分号结束，括号的使用，等等）
懂得预处理器将为你计算常数表达式的值，因此，直接写出你是如何计算一年中有多少秒而不是计算出实际的值，是更清晰而没有代价的。
意识到这个表达式将使一个16位机的整型数溢出-因此要用到长整型符号L,告诉编译器这个常数是的长整型数。
如果你在你的表达式中用到UL（表示无符号长整型），那么你有了一个好的起点。记住，第一印象很重要。
71.写一个”标准"宏MIN ，这个宏输入两个参数并返回较小的一个。
答：
1
#define?MIN(A,B)?（（A）?<=?(B)???(A)?:?(B))
这个测试是为下面的目的而设的：
标识#define在宏中应用的基本知识。这是很重要的，因为直到嵌入(inline)操作符变为标准C的一部分，宏是方便产生嵌入代码的唯一方
法，
对于嵌入式系统来说，为了能达到要求的性能，嵌入代码经常是必须的方法。
三重条件操作符的知识。这个操作符存在C语言中的原因是它使得编译器能产生比 if-then-else 更优化的代码，了解这个用法是很重要的。
懂得在宏中小心地把参数用括号括起来
我也用这个问题开始讨论宏的副作用，例如：当你写下面的代码时会发生什么事?
1
least?=?MIN(*p++,?b);
结果是：
1
((*p++)?<=?(b)???(*p++)?:?(*p++))
这个表达式会产生副作用，指针p会作三次++自增操作。
72.关键字const有什么含意？修饰类呢?static的作用,用于类呢?还有extern c的作用
答：

const 意味着"只读"，下面的声明都是什么意思?

const?int?a;
int?const?a;
const?int?*a;
int?*?const?a;
int?const?*?a?const;
前两个的作用是一样，a是一个常整型数。
第三个意味着a是一个指向常整型数的指针（也就是，整型数是不可修改的，但指针可以）。
第四个意思a是一个指向整型数的常指针（也就是说，指针指向的整型数是可以修改的，但指针是不可修改的）。
最后一个意味着a是一个指向常整型数的常指针（也就是说，指针指向的整型数是不可修改的，同时指针也是不可修改的）。
结论：
关键字const的作用是为给读你代码的人传达非常有用的信息，实际上，声明一个参数为常量是为了告诉了用户这个参数的应用目的。
如果你曾花很多时间清理其它人留下的垃圾，你就会很快学会感谢这点多余的信息。（当然，懂得用const的程序员很少会留下的垃圾让别人来清理的）?通过给优化器一些附加的信息，使用关键字const也许能产生更紧凑的代码。合理地使用关键字const可以使编译器很自然地保护那些不希望被改变的参数，防止其被无意的代码修改。简而言之，这样可以减少bug的出现。
1).欲阻止一个变量被改变，可以使用 const 关键字。在定义该 const 变量时，通常需要对它进行初始化，因为以后就没有机会再去改变它了；
2).对指针来说，可以指定指针本身为 const，也可以指定指针所指的数据为 const，或二者同时指定为 const；
3).在一个函数声明中，const 可以修饰形参，表明它是一个输入参数，在函数内部不能改变其值；
4).对于类的成员函数，若指定其为 const 类型，则表明其是一个常函数，不能修改类的成员变量；
5).对于类的成员函数，有时候必须指定其返回值为 const 类型，以使得其返回值不为“左值”。
73. 关键字volatile有什么含意?并给出三个不同的例子。
答：一个定义为 volatile的变量是说这变量可能会被意想不到地改变，这样，编译器就不会去假设这个变量的值了。精确地说就是，优化器在用到这个变量时必须每次都小心地重新读取这个变量的值，而不是使用保存在寄存器里的备份。
下面是volatile变量的几个例子：
并行设备的硬件寄存器（如：状态寄存器）
一个中断服务子程序中会访问到的非自动变量(Non-automatic variables)
多线程应用中被几个任务共享的变量
74. 一个参数既可以是const还可以是volatile吗？ 一个指针可以是volatile 吗？解释为什么。
答：1).是的。一个例子是只读的状态寄存器。它是volatile因为它可能被意想不到地改变。它是const因为程序不应该试图去修改它。
2).是的。尽管这并不很常见。一个例子是当一个中服务子程序修该一个指向一个buffer的指针时。
75 . static 关键字的作用：
答：
1).函数体内 static 变量的作用范围为该函数体，不同于 auto 变量，该变量的内存只被分配一次，因此其值在下次调用时仍维持上次的值；
2).在模块内的 static 全局变量可以被模块内所用函数访问，但不能被模块外其它函数访问；
3).在模块内的 static 函数只可被这一模块内的其它函数调用，这个函数的使用范围被限制在声明它的模块内；

4).在类中的 static 成员变量属于整个类所拥有，对类的所有对象只有一份拷贝；
5).在类中的 static 成员函数属于整个类所拥有，这个函数不接收 this 指针，因而只能访问类的static成员变量。
76. 线程与进程的区别和联系?
答：
1). 进程和线程都是由操作系统所体会的程序运行的基本单元，系统利用该基本单元实现系统对应用的并发性
2). 进程和线程的主要差别在于它们是不同的操作系统资源管理方式。
3). 进程有独立的地址空间，一个进程崩溃后，在保护模式下不会对其它进程产生影响，而线程只是一个进程中的不同执行路径。
4.)线程有自己的堆栈和局部变量，但线程之间没有单独的地址空间，一个线程死掉就等于整个进程死掉。所以多进程的程序要比多线程的程序健壮，但在进程切换时，耗费资源较大，效率要差一些。

5). 但对于一些要求同时进行并且又要共享某些变量的并发操作，只能用线程，不能用进程。
77. 列举几种进程的同步机制，并比较其优缺点。
答： 原子操作 ?信号量机制 ? ?自旋锁 ? ?管程，会合，分布式系统
78. 进程之间通信的途径
答：共享存储系统消息传递系统管道：以文件系统为基础
79. 进程死锁的原因
答：资源竞争及进程推进顺序非法
80. 死锁的4个必要条件
答：互斥、请求保持、不可剥夺、环路
81. 死锁的处理
答：鸵鸟策略、预防策略、避免策略、检测与解除死锁
82. cocoa touch框架
答：iPhone OS 应用程序的基础 Cocoa Touch 框架重用了许多 Mac 系统的成熟模式，但是它更多地专注于触摸的接口和优化。
UIKit 为您提供了在 iPhone OS 上实现图形，事件驱动程序的基本工具，其建立在和 Mac OS X 中一样的 Foundation 框架上，包括文件处理，网络，字符串操作等。
Cocoa Touch 具有和 iPhone 用户接口一致的特殊设计。有了 UIKit，您可以使用 iPhone OS 上的独特的图形接口控件，按钮，以及全屏视图的功能，您还可以使用加速仪和多点触摸手势来控制您的应用。
各色俱全的框架 除了UIKit 外，Cocoa Touch 包含了创建世界一流 iPhone 应用程序需要的所有框架，从三维图形，到专业音效，甚至提供设备访问 API 以控制摄像头，或通过 GPS 获知当前位置。
Cocoa Touch 既包含只需要几行代码就可以完成全部任务的强大的 Objective-C 框架，也在需要时提供基础的 C 语言 API 来直接访问系统。这些框架包括：
Core Animation：通过 Core Animation，您就可以通过一个基于组合独立图层的简单的编程模型来创建丰富的用户体验。
Core Audio：Core Audio 是播放，处理和录制音频的专业技术，能够轻松为您的应用程序添加强大的音频功能。
Core Data：提供了一个面向对象的数据管理解决方案，它易于使用和理解，甚至可处理任何应用或大或小的数据模型。
功能列表：框架分类
下面是 Cocoa Touch 中一小部分可用的框架：
音频和视频：Core Audio ， OpenAL ， Media Library ， AV Foundation

数据管理： Core Data， SQLite
图形和动画： Core Animation， OpenGL ES， Quartz 2D
网络：Bonjour， WebKit， BSD Sockets
用户应用：Address Book， Core Location， Map Kit， Store Kit
83. 自动释放池是什么,如何工作
答：当您向一个对象发送一个autorelease消息时，Cocoa就会将该对象的一个引用放入到最新的自动释放.它仍然是个正当的对象，因此自动释放池定义的作用域内的其它对象可以向它发送消息。当程序执行到作用域结束的位置时，自动释放池就会被释放，池中的所有对象也就被释放。
84. Objective-C的优缺点。
答：objc优点：
1). ?Cateogies
2). ?Posing
3). 动态识别
4).指标计算
5).弹性讯息传递
6).不是一个过度复杂的 C 衍生语言
7).Objective-C 与 C++ 可混合编程
objc缺点:
1).不支援命名空间
2).不支持运算符重载
3).不支持多重继承
4).使用动态运行时类型，所有的方法都是函数调用，所以很多编译时优化方法都用不到。（如内联函数等），性能低劣。
85. sprintf,strcpy,memcpy使用上有什么要注意的地方。
答:
1). sprintf是格式化函数。将一段数据通过特定的格式，格式化到一个字符串缓冲区中去。sprintf格式化的函数的长度不可控，有可能格式化后的字符串会超出缓冲区的大小，造成溢出。
2).strcpy是一个字符串拷贝的函数，它的函数原型为strcpy(char *dst, const char *src
将src开始的一段字符串拷贝到dst开始的内存中去，结束的标志符号为 '\0'，由于拷贝的长度不是由我们自己控制的，所以这个字符串拷贝很容易出错。
3). memcpy是具备字符串拷贝功能的函数，这是一个内存拷贝函数，它的函数原型为memcpy(char *dst, const char* src, unsigned int len);将长度为len的一段内存，从src拷贝到dst中去，这个函数的长度可控。但是会有内存叠加的问题。
86. readwrite，readonly，assign，retain，copy，nonatomic 属性的作用
答：@property是一个属性访问声明，扩号内支持以下几个属性：
1).getter=getterName，setter=setterName，设置setter与 getter的方法名
2).readwrite,readonly，设置可供访问级别
2).assign，setter方法直接赋值，不进行任何retain操作，为了解决原类型与环循引用问题
3).retain，setter方法对参数进行release旧值再retain新值，所有实现都是这个顺序(CC上有相关资料)
4).copy，setter方法进行Copy操作，与retain处理流程一样，先旧值release，再 Copy出新的对象，retainCount为1。这是为了减少对上下文的依赖而引入的机制。
5).nonatomic，非原子性访问，不加同步，多线程并发访问会提高性能。注意，如果不加此属性，则默认是两个访问方法都为原子型事务访问。锁被加到所属对象实例级。
87. http和scoket通信的区别。

答： http是客户端用http协议进行请求，发送请求时候需要封装http请求头，并绑定请求的数据，服务器一般有web服务器配合（当然也非绝对）。 http请求方式为客户端主动发起请求，服务器才能给响应，一次请求完毕后则断开连接，以节省资源。服务器不能主动给客户端响应（除非采取http长连接 技术）。iphone主要使用类是NSUrlConnection。
scoket是客户端跟服务器直接使用socket“套接字”进行连接，并没有规定连接后断开，所以客户端和服务器可以保持连接通道，双方 都可以主动发送数据。一般在游戏开发或股票开发这种要求即时性很强并且保持发送数据量比较大的场合使用。主要使用类是CFSocketRef。
88. TCP和UDP的区别
答： TCP全称是Transmission Control Protocol，中文名为传输控制协议，它可以提供可靠的、面向连接的网络数据传递服务。传输控制协议主要包含下列任务和功能：
* 确保IP数据报的成功传递。
* 对程序发送的大块数据进行分段和重组。
* 确保正确排序及按顺序传递分段的数据。
* 通过计算校验和，进行传输数据的完整性检查。
TCP提供的是面向连接的、可靠的数据流传输，而UDP提供的是非面向连接的、不可靠的数据流传输。
简单的说，TCP注重数据安全，而UDP数据传输快点，但安全性一般
89. 你了解svn,cvs等版本控制工具么?
答： 版本控制 svn,cvs 是两种版控制的器,需要配套相关的svn，cvs服务器。
scm是xcode里配置版本控制的地方。版本控制的原理就是a和b同时开发一个项目，a写完当天的代码之后把代码提交给服务器，b要做的时候先从服务器得到最新版本，就可以接着做。 如果a和b都要提交给服务器，并且同时修改了同一个方法，就会产生代码冲突，如果a先提交，那么b提交时，服务器可以提示冲突的代码，b可以清晰的看到，并做出相应的修改或融合后再提交到服务器。
90. 什么是push。
答： 客户端程序留下后门端口，客户端总是监听针对这个后门的请求，于是 服务器可以主动像这个端口推送消息。
91. 静态链接库
答：此为.a文件，相当于java里的jar包，把一些类编译到一个包中，在不同的工程中如果导入此文件就可以使用里面的类，具体使用依然是#import “ xx.h”。
92. fmmpeg框架
答： 音视频编解码框架，内部使用UDP协议针对流媒体开发，内部开辟了六个端口来接受流媒体数据，完成快速接受之目的。
93. fmdb框架
答：数据库框架，对sqllite的数据操作进行了封装，使用着可把精力都放在sql语句上面。
94. 320框架
答： ui框架，导入320工程作为框架包如同添加一个普通框架一样。cover(open) ?flower框架 (2d 仿射技术)，内部核心类是CATransform3D.
94. 什么是沙盒模型？哪些操作是属于私有api范畴?
答：某个iphone工程进行文件操作有此工程对应的指定的位置，不能逾越。
iphone沙箱模型的有四个文件夹documents，tmp，app，Library，永久数据存储一般放documents文件夹，得到模拟器的路径的可使用NSHomeDirectory()方法。Nsuserdefaults保存的文件在tmp文件夹里。
95. 在一个对象的方法里面：self.name= “object”；和 name =”object” 有什么不同吗?
答：self.name =”object”：会调用对象的setName()方法；
name = “object”：会直接把object赋值给当前对象的name属性。

96. 请简要说明viewDidLoad和viewDidUnload何时调用
答：viewDidLoad在view从nib文件初始化时调用，loadView在controller的view为nil时调用。此方法在编程实现view时调用，view控制器默认会注册memory warning notification，当view controller的任何view没有用的时候，viewDidUnload会被调用，在这里实现将retain的view release，如果是retain的IBOutlet view 属性则不要在这里release，IBOutlet会负责release 。
97. 简述内存分区情况
答：
1).代码区：存放函数二进制代码
2).数据区：系统运行时申请内存并初始化，系统退出时由系统释放。存放全局变量、静态变量、常量
3).堆区：通过malloc等函数或new等操作符动态申请得到，需程序员手动申请和释放
4).栈区：函数模块内申请，函数结束时由系统自动释放。存放局部变量、函数参数
98. 队列和栈有什么区别：
答：队列和栈是两种不同的数据容器。从”数据结构”的角度看，它们都是线性结构，即数据元素之间的关系相同。
队列是一种先进先出的数据结构，它在两端进行操作，一端进行入队列操作，一端进行出列队操作。

栈是一种先进后出的数据结构，它只能在栈顶进行操作，入栈和出栈都在栈顶操作。
99. HTTP协议中，POST和GET的区别是什么?
答：
1).GET 方法
GET 方法提交数据不安全，数据置于请求行，客户端地址栏可见;
GET 方法提交的数据大小有限
GET 方法不可以设置书签
2).POST 方法
POST 方法提交数据安全，数据置于消息主体内，客户端不可见
POST 方法提交的数据大小没有限制
POST 方法可以设置书签
100. ?iOS的系统架构
答： iOS的系统架构分为（ 核心操作系统层 theCore OS layer ）、（ 核心服务层theCore Services layer ）、（ 媒体层 theMedia layer ）和（ Cocoa 界面服务层 the Cocoa Touch layer ）四个层次。
101. ?控件主要响应3种事件
答：1). 基于触摸的事件；?2). 基于值的事件；?3).基于编辑的事件。
102. ?xib文件的构成分为哪3个图标? 都具有什么功能。
答： File’s Owner 是所有 nib 文件中的每个图标，它表示从磁盘加载 nib 文件的对象；
First Responder 就是用户当前正在与之交互的对象；
View 显示用户界面；完成用户交互；是 UIView 类或其子类。
103. ?简述视图控件器的生命周期。
答： loadView 尽管不直接调用该方法，如多手动创建自己的视图，那么应该覆盖这个方法并将它们赋值给试图控制器的 view 属性。
viewDidLoad 只有在视图控制器将其视图载入到内存之后才调用该方法，这是执行任何其他初始化操作的入口。
viewDidUnload 当试图控制器从内存释放自己的方法的时候调用，用于清楚那些可能已经在试图控制器中创建的对象。

viewVillAppear 当试图将要添加到窗口中并且还不可见的时候或者上层视图移出图层后本视图变成顶级视图时调用该方法，用于执行诸如改变视图方向等的操作。实现该方法时确保调用 [super viewWillAppear:
viewDidAppear 当视图添加到窗口中以后或者上层视图移出图层后本视图变成顶级视图时调用，用于放置那些需要在视图显示后执行的代码。确保调用 [super viewDidAppear：] 。
104. ?动画有基本类型有哪几种；表视图有哪几种基本样式。
答：动画有两种基本类型：隐式动画和显式动画。
105. ?实现简单的表格显示需要设置UITableView的什么属性、实现什么协议?
答：实现简单的表格显示需要设置 UITableView 的 dataSource 和 delegate 属性，实现 UITableViewDataSource 和 UITableViewDelegate 协议。
106. ?Cocoa Touch提供了哪几种Core Animation过渡类型?
答： Cocoa Touch 提供了 4 种 Core Animation 过渡类型，分别为：交叉淡化、推挤、显示和覆盖。

107. ?UIView与CLayer有什么区别?
答:
1).UIView 是 iOS 系统中界面元素的基础，所有的界面元素都是继承自它。它本身完全是由 CoreAnimation 来实现的。它真正的绘图部分，是由一个 CALayer 类来管理。 UIView 本身更像是一个 CALayer 的管理器，访问它的跟绘图和跟坐标有关的属性。
2).UIView 有个重要属性 layer ，可以返回它的主 CALayer 实例。
3).UIView 的 CALayer 类似 UIView 的子 View 树形结构，也可以向它的 layer 上添加子layer ，来完成某些特殊的表示。即 CALayer 层是可以嵌套的。
4).UIView 的 layer 树形在系统内部，被维护着三份 copy 。分别是逻辑树，这里是代码可以操纵的；动画树，是一个中间层，系统就在这一层上更改属性，进行各种渲染操作；显示树，其内容就是当前正被显示在屏幕上得内容。
5).动画的运作：对 UIView 的 subLayer （非主 Layer ）属性进行更改，系统将自动进行动画生成，动画持续时间的缺省值似乎是 0.5 秒。
6).坐标系统： CALayer 的坐标系统比 UIView 多了一个 anchorPoint 属性，使用CGPoint 结构表示，值域是 0~1 ，是个比例值。这个点是各种图形变换的坐标原点，同时会更改 layer 的 position 的位置，它的缺省值是 {0.5,0.5} ，即在 layer 的中央。
7).渲染：当更新层，改变不能立即显示在屏幕上。当所有的层都准备好时，可以调用 setNeedsDisplay 方法来重绘显示。
8).变换：要在一个层中添加一个 3D 或仿射变换，可以分别设置层的 transform 或affineTransform 属性。
9).变形： Quartz Core 的渲染能力，使二维图像可以被自由操纵，就好像是三维的。图像可以在一个三维坐标系中以任意角度被旋转，缩放和倾斜。 CATransform3D 的一套方法提供了一些魔术般的变换效果。
108. Quatrz 2D的绘图功能的三个核心概念是什么并简述其作用。
答：上下文：主要用于描述图形写入哪里；
路径：是在图层上绘制的内容；
状态：用于保存配置变换的值、填充和轮廓， alpha 值等。
109. ?iPhone OS主要提供了几种播放音频的方法?
答： SystemSound Services
AVAudioPlayer 类
Audio Queue Services
OpenAL
110. ?使用AVAudioPlayer类调用哪个框架、使用步骤?

答： AVFoundation.framework
步骤：配置 AVAudioPlayer 对象；
实现 AVAudioPlayer 类的委托方法；
控制 AVAudioPlayer 类的对象；
监控音量水平；
回放进度和拖拽播放。
111. ?有哪几种手势通知方法、写清楚方法名?
答：
-(void)touchesBegan:(NSSet*)touchedwithEvent:(UIEvent*)event;
-(void)touchesMoved:(NSSet*)touched withEvent:(UIEvent*)event;
-(void)touchesEnded:(NSSet*)touchedwithEvent:(UIEvent*)event;
-(void)touchesCanceled:(NSSet*)touchedwithEvent:(UIEvent*)event;
112. ?CFSocket使用有哪几个步骤。
答：创建 Socket 的上下文；创建 Socket ； 配置要访问的服务器信息；封装服务器信息；连接服务器；
113. ?Core Foundation中提供了哪几种操作Socket的方法?
答： CFNetwork 、 CFSocket 和 BSD Socket 。
114. ?解析XML文件有哪几种方式?
答：以 DOM 方式解析 XML 文件；以 SAX 方式解析 XML 文件；
115. ios 平台怎么做数据的持久化?coredata 和sqlite有无必然联系？coredata是一个关系型数据库吗?
答：iOS 中可以有四种持久化数据的方式：属性列表(plist)、对象归档、 SQLite3 和 Core Data；
core data 可以使你以图形界面的方式快速的定义 app 的数据模型，同时在你的代码中容易获取到它。 coredata 提供了基础结构去处理常用的功能，例如保存，恢复，撤销和重做，允许你在 app 中继续创建新的任务。在使用 core data 的时候，你不用安装额外的数据库系统，因为 core data 使用内置的 sqlite 数据库。 core data 将你 app 的模型层放入到一组定义在内存中的数据对象。
coredata 会追踪这些对象的改变，同时可以根据需要做相反的改变，例如用户执行撤销命令。当 core data 在对你 app 数据的改变进行保存的时候， core data 会把这些数据归档，并永久性保存。
mac os x 中sqlite 库，它是一个轻量级功能强大的关系数据引擎，也很容易嵌入到应用程序。可以在多个平台使用， sqlite 是一个轻量级的嵌入式 sql 数据库编程。与 core data 框架不同的是，
sqlite 是使用程序式的， sql 的主要的 API 来直接操作数据表。 Core Data 不是一个关系型数据库，也不是关系型数据库管理系统 (RDBMS) 。虽然 Core Dta 支持SQLite 作为一种存储类型，但它不能使用任意的 SQLite 数据库。 Core Data 在使用的过程种自己创建这个数据库。 Core Data 支持对一、对多的关系。
116. ?tableView 的重用机制?
答：UITableView 通过重用单元格来达到节省内存的目的: 通过为每个单元格指定一个重用标识符(reuseIdentifier),即指定了单元格的种类,以及当单元格滚出屏幕时,允许恢复单元格以便重用.对于不同种类的单元格使用不同的ID,对于简单的表格,一个标识符就够了.