能力要求：精通iphone的UI开发，能熟练操作复杂表视图，熟练使用图层技术， 可以自定义UI控件，使用类别扩展系统控件功能;

擅长通讯技术，熟悉各种通信协议，精通xml, json, 二进制或其他形式的自定义解析，能架设服务器实现客户端与服务器的通讯以提交开发效率;

熟练掌握各种数据存储技术，如core data, sqlite, 对象序列化，文件读写操作，熟悉数据库的设计。

精通 object-c，java， c 等编程语言, 熟悉c++，对于 面向对象编程思想有深入理解，熟悉常见设计模式的应用，

1.objective-c中的数字对象都有哪些，简述它们与基本数据类型的区别是什么

2.用NSLog函数输出一个浮点类型，结果四舍五入，并保留一位小数

3.截取字符串”20｜http://www.621life.com“ 中 ‘｜’字符前面及后面的数据，分别输出它们

4.objective-c中的词典对象、可变词典对象是哪个，初始化一个含有两个键值对的可变词典对象，并动态的添加和删除一条记录，输出第一条记录

5.获取项目根路径，并在其下创建一个名称为userData的目录。

6.在一个对象的方法里面:self.name = “object”;和name ＝”object”有什么不同吗?

7.定义属性时，什么情况使用copy，assign，和retain

8.ViewController 的viewDidLoad,viewWillAppear,viewDidUnload,dealloc

分别是在什么时候调用，在自定义ViewController的时候这几个函数里面应该做什么工作？

9.简述objective-c内存管理的实现机制，并简述什么时候由你负责释放对象，什么时候不由你释放

10.类的定义及声明文件以什么为后缀名？

11.怎样自动生成属性的获取方法和设置方法

12.声明一个静态方法和一个实例方法

13.写一个发送同步http请求，并获得返回结果的方法

14.怎样启动一个新线程，子线程怎样刷新主UI

15.什么是MVC，你工作时怎样运用它

16. 如何判断一个对象是否属于某个类?

第1题：

NSNumber是Objective-c的数字对象，相当于将一个数字包装为一个对象。

你可以利用如下方式建立数字对象或者从对象中拿出基本类型的数据

NSNumber \*number = [NSNumber numberWithInt:123]; //将int型的123转成NSNumber

int a= [number intValue]; //得到int类型的值

注意区分NSInteger、CGFloat这些都是在Foundation框架下的别名，点进去看实际上它们还是int 和 float

第2题：

NSLog(@”%.1f”,(NSNumber \*)[@”48.79”, doublueValue])

第3题：

NSRange range = [responseString rangeOfString:@"|"];

int location = range.location;

NSString \*str1 = [responseString substringToIndex:location];

NSString \*str2 = [responseString substringFromIndex:location+1];

第4题：

字典对象：NSDictionary 和NSMutableDictionary

NSMutableDictionary\*dic=[NSMutableDictionary dictionaryWithObjectsAndKeys:@"value1",@"key1",@"value2",@"key2",nil];

[dic setObject:@"value3" forKey:@"key3"];

[dic removeObjectForKey:@"key3"];

[dic objectForKey:@"key1"];

第5题：

// 获取根路径

NSArray \* paths=NSSearchPathForDirectoriesInDomains

(NSDocumentDirectory,NSUserDomainMask, YES);

NSString \*documentsDirectory = [paths objectAtIndex:0];// 创建文件系统管理器

NSFileManager \*fileManager = [[NSFileManager alloc] init];

// 判断userData目录是否存在

if( ![fileManager fileExistsAtPath:[NSString stringWithFormat:@"%@/userData", documentsDirectory]]) {

// 不存在,创建一个userData目录

[fileManager createDirectoryAtPath:[NSString stringWithFormat:@"%@/userData", documentsDirectory] withIntermediateDirectories:false attributes:nil error:nil];

}

第6题：

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。

并且 self.name 这样retainCount会加１,而name就不会。

第 7题：

assign用于简单数据类型，如NSInteger,double,bool。 retain 和copy用于对象，copy用于当 a指向一个对象，b也想指向同样的对象的时候。如果用assign，a如果释放，再调用b会crash,如果用copy 的方式，a和b各自有自己的内存，就可以解决这个问题。retain 会使计数器加一，也可以解决assign的问题。另外：tomic和nonatomic用来决定编译器生成的getter和setter是否为原子操作。在多线程环境下，原子操作是必要的，否则有可能引起错误的结果。

第8题：

init:方法

在init方法中实例化必要的对象（遵从LazyLoad思想）

init方法中初始化ViewController本身

loadView:方法

这是当没有正在使用nib视图页面，子类将会创建自己的自定义视图层。绝不能直接调用。

如果手工维护views，必须重载重写该方法。

当view需要被展示而它却是nil时，viewController会调用该方法。

loadView和IB构建view

你在控制器中实现了loadView方法，那么你可能会在应用运行的某个时候被内存管理控制调用。 如果设备内存不足的时候， view 控制器会收到didReceiveMemoryWarning的消息。 默认的实现是检查当前控制器的view是否在使用。 如果它的view不在当前正在使用的view hierarchy里面，且你的控制器实现了loadView方法，那么这个view将被release,

loadView方法将被再次调用来创建一个新的view。

viewDidLoad:方法

在视图加载后被调用：

如果是在代码中创建的视图加载器，他将会在loadView方法后被调用；

如果是从nib视图页面输出，他将会在视图设置好后后被调用。

重载重写该方法以进一步定制view

在iPhone OS 3.0及之后的版本中，还应该重载重写viewDidUnload来释放对view的任何索引

viewDidLoad后调用数据Model

viewDidUnload:方法

当系统内存吃紧的时候会调用该方法（注：viewController没有被dealloc）

内存吃紧时，在iPhone OS 3.0之前didReceiveMemoryWarning是释放无用内存的唯一方式，但是OS 3.0及以后viewDidUnload方法是更好的方式

在该方法中将所有IBOutlet（无论是property还是实例变量）置为nil（系统release view时已经将其release掉了）

在该方法中释放其他与view有关的对象、其他在运行时创建（但非系统必须）的对象、在viewDidLoad中被创建的对象、缓存数据等 release对象后，将对象置为nil（IBOutlet只需要将其置为nil，系统release view时已经将其release掉了）

一般认为viewDidUnload是viewDidLoad的镜像，因为当view被重新请求时，viewDidLoad还会重新被执行

viewDidUnload中被release的对象必须是很容易被重新创建的对象（比如在viewDidLoad或其他方法中创建的对象），不要release用户数据或其他很难被重新创建的对象

dealloc:方法

viewDidUnload和dealloc方法没有关联，dealloc还是继续做它该做的事情

viewWillAppear: 方法

Called when the view is about to made visible. Default does nothing

视图即将可见时调用。默认情况下不执行任何操作

viewDidAppear:

Called when the view has been fully transitioned onto the screen. Default does nothing

视图已完全过渡到屏幕上时调用

viewWillDisappear:

Called when the view is dismissed, covered or otherwise hidden. Default does nothing

视图被驳回时调用，覆盖或以其他方式隐藏。默认情况下不执行任何操作

viewDidDisappear:

Called after the view was dismissed, covered or otherwise hidden. Default does nothing

视图被驳回后调用，覆盖或以其他方式隐藏。默认情况下不执行任何操作

第9题：

1.Objective-C中所有对象都在堆区建立，由程序员负责释放对象所占用的内存。内存管理机制由3种：垃圾回收、引用计数、C语言方式。

2.垃圾回收是Mac OS10.5提供的新方案，在系统存在一个垃圾收集器。如果发现某个对象没有被任何对象使用，该对象被自动释放。

3.C语言方式，原始内存管理方式。用户手动调用malloc、calloc函数分配内存，free回收内存。

4.引用计数机制：对象创建后，运行时系统通过对象维护的一个计数器来描述有多少个其他对象在使用自己，当计数器为0时，释放该对象占用的内存空间（该对象调用dealloc方法）。

5,内存管理规则：当使用alloc，new或copy创建一个对象时，对象的引用计数被设置为1.；向对象发送retain消息，对象引用计数加1；向对象发送release消息时，对象引用计数减1；当对象引用计数为0时，运行时系统向对象发送dealloc消息并回收对象所占用的内存。

6.注意：如果类的实例变量是对象指针，则必须重载dealloc方法（例如圆circle类：有一个变量point是Point\*类型）；dealloc方法不能直接调用，只有对象的引用计数为0的时候，系统会自动向对象发送dealloc消息。

内存管理原则

1. 谁创建，谁释放（“谁污染，谁治理”）。如果你通过alloc、new或者(mutable)copy来创建一个对象，那么你必须调用release或autorelease。或句话说，不是你创建的，就不用你去释放

2. 一般来说，除了alloc、new或copy之外的方法创建的对象都被声明了autorelease（autorelease是延迟释放内存，不用你自己去手动释放，系统会知道在什么时候该去释放掉它。）

3. 谁retain，谁release。只要你调用了retain，无论这个对象是如何生成的，你都要调用release

第10题：

.h:头文件。头文件包含类，类型，函数和常数的声明。

.m ：源代码文件。这是典型的源代码文件扩展名，可以包含Objective-C和C代码。

.mm ：源代码文件。带有这种扩展名的源代码文件，除了可以包含Objective-C和C代码以外还可以包含C++代码。

仅在你的Objective-C代码中确实需要使用C++类或者特性的时候才用这种扩展名

.cpp：只能编译C++

当你需要在源代码中包含头文件的时候，你可以使用标准的#include编译选项，但是Objective-C提供了更好的方法。#import选项和#include选项完全相同，只是它可以确保相同的文件只会被包含一次。Objective-C的例子和文档都倾向于使用#import。

.m 和.mm 的区别是告诉gcc 在编译时要加的一些参数。

第11题：

@property @synthesize

第12题：

静态方法：类方法 +

实例方法：对象方法 –

第13题：

// 初始化请求

NSMutableURLRequest \*request = [[NSMutableURLRequest alloc] init];

// 设置URL

[request setURL:[NSURL URLWithString:urlStr]];

// 设置HTTP方法

[request setHTTPMethod:@"GET"];

// 发送同步请求, returnData就是返回的数据

NSData \*returnData = [NSURLConnection sendSynchronousRequest:request

returningResponse:nil error:nil];

//得到返回结果以String形式输出

NSString\* strRet = [[NSString alloc] initWithData:data encoding:NSUTF8String];

NSLog(strRet);

[request release];

第14题：

启动一个线程：

1. NSThread的创建主要有两种直接方式：

[NSThread detachNewThreadSelector:@selector(myThredaMethod:) toTarget:self withObject:nil];

和

NSThread \* myThread =[[NSThread alloc] initWithTarget:self selector:@selector(myThredaMethod:) object:nil];

[myThread start];

2.使用NSObject

使用NSObject直接就加入了对多线程的支持，允许对象的某个方法在后台运行。

[my0bj performSelectorInBackground:@selector（doSomething） withObject:nil];

3. 声明一个NSOperationQueue

NSOperationQueue \*queue = [[[NSOperationQueue alloc ] init] autorelease];

[queue addOperation:testNSoperation];

它会自动调用TestNSOperation里的start函数，如果需要多个NSOperation,你需要设置queue的一些属性，如果多个NSOperation之间有依赖关系，也可以设置，具体可以参考API文档。

非并发执行

-(void)main

只需要重载这个main方法就可以了。

4. GCD

dispatch\_async(getDataQueue,^{

//获取数据,获得一组后,刷新UI.

dispatch\_aysnc(mainQueue, ^{

//UI的更新需在主线程中进行

};

}

)；

方法1：performSelectorOnMainThread

[self performSelectorOnMainThread:@selector(updateUI:) withObject:image waitUntilDone:NO];

方法2：

dispatch\_async(dispatch\_get\_main\_queue(), ^{ ... })

dispatch\_async(dispatch\_get\_global\_queue(0, 0), ^{

// 处理耗时操作的代码块...

//通知主线程刷新

dispatch\_async(dispatch\_get\_main\_queue(), ^{

//回调或者说是通知主线程刷新，

});

});

dispatch\_async开启一个异步操作，第一个参数是指定一个gcd队列，第二个参数是分配一个处理事物的程序块到该队列。

dispatch\_get\_global\_queue(0, 0)，指用了全局队列。

一般来说系统本身会有3个队列。global\_queue，current\_queue,以及main\_queue.

获取一个全局队列是接受两个参数，第一个是我分配的事物处理程序块队列优先级。分高低和默认，0为默认2为高，-2为低

处理完事物后，需要将结果返回给或者是刷新UI主线程，同样，和上面一样，抓取主线程，程序块操作。

第15题：

模型对象

模型对象封装了应用程序的数据，并定义操控和处理该数据的逻辑和运算。例如，模型对象可能是表示游戏中的角色或地址簿中的联系人。用户在视图层中所进行的创建或修改数据的操作，通过控制器对象传达出去，最终会创建或更新模型对象。模型对象更改时（例如通过网络连接接收到新数据），它通知控制器对象，控制器对象更新相应的视图对象。

视图对象

视图对象是应用程序中用户可以看见的对象。视图对象知道如何将自己绘制出来，并可能对用户的操作作出响应。视图对象的主要目的，就是显示来自应用程序模型对象的数据，并使该数据可被编辑。尽管如此，在 MVC 应用程序中，视图对象通常与模型对象分离。

在iOS应用程序开发中，所有的控件、窗口等都继承自 UIView，对应MVC中的V。UIView及其子类主要负责UI的实现，而UIView所产生的事件都可以采用委托的方式，交给UIViewController实现。

控制器对象

在应用程序的一个或多个视图对象和一个或多个模型对象之间，控制器对象充当媒介。控制器对象因此是同步管道程序，通过它，视图对象了解模型对象的更改，反之亦然。控制器对象还可以为应用程序执行设置和协调任务，并管理其他对象的生命周期。

控制器对象解释在视图对象中进行的用户操作，并将新的或更改过的数据传达给模型对象。模型对象更改时，一个控制器对象会将新的模型数据传达给视图对象，以便视图对象可以显示它。

对于不同的UIView，有相应的UIViewController，对应MVC中的C。例如在iOS上常用的UITableView，它所对应的Controller就是UITableViewController。

Model和View永远不能相互通信，只能通过Controller传递。

Controller可以直接与Model对话（读写调用Model），Model通过Notification和KVO机制与Controller间接通信。

Controller可以直接与View对话，通过outlet,直接操作View,outlet直接对应到View中的控件,View通过action向Controller报告事件的发生(如用户Touch我了)。Controller是View的直接数据源（数据很可能是Controller从Model中取得并经过加工了）。Controller是View的代理（delegate),以同步View与Controller。

第16题：

respondsToSelector ,performSelector,conformstoProrocol,isKindofClass,isMemberofClass

17.object-c如何实现私有方法?

答:在object-c中没有私有方法,只有静态方法和实力方法.但是可以用@private修饰私有变量.用.m文件中使用catogory来实现私有方法

18.BOOL与bool区别?

BOOL 只是一个宏，不是基本的数据类型，BOOL实际上=int，占4个字节空间。

bool 是C的基础数据类型，占用1个字节。

对于BOOL类型，当值为1时为真，即TRUE；当值为0时为假，即FALSE；

FALSE和TRUE都是宏。

19.用c语言写一个函数判断两个链表(单链表不存在环)是否相交?

bool IsExitsLoop(slist \* head){ slist \* slow = head , \* fast = head; while (fast && fast -> next)

{

slow = slow -> next;fast = fast -> next -> next;if ( slow == fast ) break ;

}

return ! (fast == NULL || fast -> next == NULL);}

20.线程和进程的区别和联系?

答: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。

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。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。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。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。

21.简单阐述堆和栈的区别?

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。

1.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时示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。

2.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

3.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。

4.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。

22.为什么很多内置类如UITableViewController的delegate属性都是assign而不是retain的?

答：会引起循环引用,所有的引用计数系统，都存在循环应用的问题。例如下面的引用关系：

\* 对象a创建并引用到了对象b.

\* 对象b创建并引用到了对象c.

\* 对象c创建并引用到了对象b.

这时候b和c的引用计数分别是2和1。

当a不再使用b，调用release释放对b的所有权，因为c还引用了b，所以b的引用计数为1，b不会被释放。

b不释放，c的引用计数就是1，c也不会被释放。从此，b和c永远留在内存中。

这种情况，必须打断循环引用，通过其他规则来维护引用关系。我们常见的delegate往往是assign方式的属性而不是retain方式的属性，

赋值不会增加引用计数，就是为了防止delegation两端产生不必要的循环引用。

如果一个UITableViewController 对象a通过retain获取了UITableView对象b的所有权，这个UITableView对象b的delegate又是a，

如果这个delegate是retain方式的，那基本上就没有机会释放这两个对象了。自己在设计使用delegate模式时，也要注意这点。

23.#import跟#include的区别?

答:前者不会引起交叉编译的问题.因为在object-c中会存在c/c++和object-c混编的问题,如果用include引入头文件,会导致交叉编译.

24.请列举你熟悉cocoa touch框架(至少三个)

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

25.ViewController 的 didReceiveMemoryWarning 是在什么时候被调用的？

内存警告会调用didReceiveMemoryWarning

答:相机会出现内存不足的情况 去掉调用

[super didReceiveMemoryWarning];

26.objective-c中是所有对象间的交互是如何实现的?

答:通过代理去进行通讯，或者通过观察者消息模式,Blocks ,delegate

27.Sizeof与strlen的区别和联系？

sizeof()()用来计算内存的strlen用来计算字符长度的。来个具体点的，一看就知道了。char a[] = "hello";sizeof()(a) = 6;strlen(a) = 5;sizeof()会计算字符结尾的\0strlen 不会计算\0sizeof操作符返回占的空间；而对于一个数组，sizeof返回这个数组所有元素占的总空间。

而strlen不区分是数组还是指针，就读到\0为止返回长度。而且strlen是不把\0计入字符串的长度的。

28.文件处理用哪些类（写1－2个）?

答:NSFileManager\NSUserDefaults open, fopen

29.写出归档相关类 ?

答:NSKeyedArchiver、NSKeyedUnarchiver

30.如何实现Push 通知的机制（大致写下）?

答:1.用网站把要发送的消息、目的iPhone的标识打包，发给APNS。

2.APNS在自身的已注册Push服务的iPhone列表中，查找有相应标识的iPhone，并把消息发到iPhone。

3.iPhone把发来的消息传递给相应的应用程序， 并且按照设定弹出Push通知

31.何实现滚动到底部自动加载，以及显示一个 loading cell?

答:用scrollViewDidEndDragging:(UIScrollView \*)scrollView willDecelerate:(BOOL)decelerate方法计算是否触底，调用异步刷新事件scrollViewDidScroll

32.如何实现从外部打开程序?

在项目的Info.plist文件中 添加一个 URL types 节点就可以了

[[UIApplication sharedApplication] openURL:[NSURL URLWithString:@"http://www.baidu.com"]];

[[UIApplication sharedApplication] openURL:[NSURL URLWithString:@"tel://18533221222"]];

[[UIApplication sharedApplication] openURL:[NSURL URLWithString:@"sms://18533221222"]];

33.代理和通知作用与区别?

答:通知需要有一个通知中心:NSNotificationCenter,自定义通知的话需要给一个名字,然后监听;优点:通知的发送者和接受者都不需要知道对方.可以指定接收通知的具体方法.通知名可以是任何字符串.缺点:较键值观察(KVO)需要多点代码,在删除前必须移除监听者.

代理:通过setDelegate来设置代理对象,最经典的例子是常用的TableView.优点:支持它的类有详尽和具体信息.缺点:该类必须支持委托.某一个时间只能有一个委托连接到某一个对象.

代理通常用于一对一的关系，通知可以是一对一或一对多或一对无的关系。代理的receive可以把结果返回给sender，通知的sender只负责把消息发送出去不需要返回结果

34.怎么读取本地文件内容?

NSString \*path = [[NSBundle mainBundle]pathForResource:@"Util-Info" ofType:@"plist"];

NSDictionary \*dic = [NSDictionary dictionaryWithContentsOfFile:path];

35.用OC写一个单例

+(id)shareDefault

{

Static id s;

If(s==nil)

{

S=[[self alloc] init]

}

Return s;

}

36.WebSocket 协议

WebSocket 协议本质上是一个基于 TCP 的协议。为了建立一个 WebSocket 连接，客户端浏览器首先要向服务器发起一个 HTTP 请求，这个请求和通常的 HTTP 请求不同，包含了一些附加头信息，其中附加头信息”Upgrade: WebSocket”表明这是一个申请协议升级的 HTTP 请求，服务器端解析这些附加的头信息然后产生应答信息返回给客户端，客户端和服务器端的 WebSocket 连接就建立起来了，双方就可以通过这个连接通道自由的传递信息，并且这个连接会持续存在直到客户端或者服务器端的某一方主动的关闭连接。