1、OC中iskindofclass与ismemberofclass的区别?

答:-(BOOL) isKindOfClass: classObj判断是否是这个类或者这个类的子类的实例；

-(BOOL) isMemberOfClass: classObj 判断是否是这个类的实例。

2、OC中实例方法与类方法的区别？

答:实例方法 调用方式[对象 方法]只有实例创建的对象可以调用，方法开头用-；

类方法 调用方式[类名 方法]只有类本身可以调用，方法开头用+。

3、分类中可以添加属性吗？

答:不可以。可以通过runtime实现。

4、下面两段代码都在主线程中运行，我们在看别人代码时会发现有时会直接调用，有时会利用performSelector调用，请说明一下二者区别，  
[delegate imageDownloader:self didFinishWithImage:image];  
[delegate performSelector:@selector(imageDownloader:didFinishWithImage:)withObject:self withObject:image]？

答:1、performSelector是运行时系统负责去找方法的，在编译时不做任何校验；如果直接调用方法编译时会自动校验。如果imageDownloader:self didFinishWithImage:image不存在，那么在直接调用时编译的时候就会发现，而如果使用performSelector的话只能在运行时才能发现;

2、直接调用方法时候，一定要在头文件中声明该方法的使用，也要将头文件import进来。而使用performSelector时候，可以不用import头文件包含方法的对象，直接用performSelector调用即可。

5、runtime中class\_addMethod方法什么时候调用?

答:当需要继承某一个类时,但是父类中并没有提供我需要的方法，而我又不知道父类中的某些方法的具体实现；或者我需要为这个类写一个分类，但是在这个分类中，我可能需要替换或者新增某个方法（分类中最好不要重写方法）时，推荐使用class\_addMethod来实现我们想要的效果。

6、多线程中堆与栈是公有还是私有的？

答：栈是私有的，堆是共有的

7、简述继承的运用场景和优缺点。

答：运用场景：

1）当有两个或两个以上的类拥有共同属性和方法时，我们可以把共有的属性方法提取到一个单独的类中，这些类继承于这个类

2）当A类拥有B类的部分属性和方法时，可以考虑用B类继承A类

3）当B是A的一种是，用继承，即B继承A

缺点：耦合性太强了，拥有继承关系的两个类之间的联系太过紧密，当一个类变动或者故障时，子类可能会发生改变或者崩溃。

优点：抽出了重复的代码，增加了类与类之间的联系。

8、简述#define定义的宏与const定义的常量的区别？

答:1）编译器处理的方式不同

define宏是在预编译处理阶段展开的

const常量是在编译运行阶段使用的

2）类型安全检查不同

define定义的宏没有类型，只是单纯的代码段，展开代码

const定义的常量具有类型，在编译阶段会进行类型检查。

3）存储方式不同

define定义的宏是展开，使用几次就展开几次，不会分配内存，但是变量定义会分配内存（#define P 2500），定义的常量会在内存中有多个拷贝，程序运行时并不分配内存，在预编译阶段已经展开代码了

const 常量会在内存中分配（只分配一次，存放在静态去），可以节省空间

1. UIView和UILayer的区别与联系？

答:

1）UIView继承与UIRespose可以响应事件，而UILayer继承于NSObject没有响应事件。

2）UIView侧重于内容的显示，UILayer侧重于图形和界面的绘制

UIView在展示的时候，View是Layer的CALayerDelegate，View展示的内容是由Calayer显示的。

3）UIView的属性修改，不会引起动画效果，但是对于CAlayer的属性修改，是有默认动画的。

4）UIView的内容展示依赖于CALayer对内容的绘制,UIView的frame也是由CALayer绘制的。

10、**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:#define MIN(x,y)((x)>(y)?(y):(x))

注意：define只会是纯替换作用，所以X，Y均需要加括号，以防止X，Y为表达式的情况,如3\*MIN（x,y）=》3\*((x)>(y)?(y):(x)),不加括号就会出现问题了。

11、readwrite、readonly、strong、copy、weak、nonatomic各有什么区别？

12、cocoapods知识集

1. pod 'AFNetworking'      //不显式指定依赖库版本，表示每次都获取最新版本
2. pod 'AFNetworking', '2.0'     //只使用2.0版本
3. pod 'AFNetworking', '> 2.0'     //使用高于2.0的版本
4. pod 'AFNetworking', '>= 2.0'     //使用大于或等于2.0的版本
5. pod 'AFNetworking', '< 2.0'     //使用小于2.0的版本
6. pod 'AFNetworking', '<= 2.0'     //使用小于或等于2.0的版本
7. pod 'AFNetworking', '~> 0.1.2'     //使用大于等于0.1.2但小于0.2的版本
8. pod 'AFNetworking', '~>0.1'     //使用大于等于0.1但小于1.0的版本
9. pod 'AFNetworking', '~>0'     //高于0的版本，写这个限制和什么都不写是一个效果，都表示使用最新版本