答案:

1. 说说下面这两种代理属性的异同点
assign: 简单赋值，不更改索引计数
对基础数据类型 （例如 NSInteger，CGFloat）和 C 数据类型（int, float, double, char, 等） 适用简单数据类型
此标记说明设置器直接进行赋值，这也是默认值。在使用垃圾收集的应用程序中，如果你要一个属性使用 assign，且这个类符合 NSCopying 协 议，你就要明确指出这个标记，而不是简单地使用默认值，否则的话，你将得到一个编译警告。这再次向编译器说明你确实需要赋值，即使它是可拷贝的。
weak 比 assign 多了一个功能，当对象消失后自动把指针变成 nil.(回答不出这句本题不得分) 10 分

2. 定义一个 blocktypedef void(^AlipayCallBack)(void).
block 使用 copy 是从 MRC 遗留下来的“传统”,在 MRC 中,方法内部的 block 是在栈区的,使用 copy 可以把它放到堆区.在 ARC 中写不写都行：对于 block 使用 copy 还是 strong 效果是一样的，但写上 copy 也无伤大雅，还能时刻提醒我们：编译器自动对 block 进行了 copy 操作。(回答这一部分即可) 10 分
copy 此特质所表达的所属关系与 strong 类似。然而设置方法并不保留新值，而是将其“拷贝” (copy)。 当属性类型为 NSString 时，经常用此特质来保护其封装性，因为传递给设置方法的新值有可能指向一个 NSMutableString 类的实例。这个类是 NSString 的子类，表示一种可修改其值的字符串，此时若是不拷贝字符串，那么设置完属性之后，字符串的值就可能会在对象不知情的情况下遭人更改。所以，这时就要拷贝一份“不可变”(immutable)的字符串，确保对象中的字符串值不会无意间变动。只要实现属性所用的对象是“可变的”(mutable)，就应该在设置新属性值时拷贝一份。(回答这一部分加分) +5 分

3. 单例的几种写法.
1) 2 分

```
+ (id)sharedInstance {
    static testClass *sharedInstance = nil;
    if (!sharedInstance) {
        sharedInstance = [[self alloc] init];
    }
    return sharedInstance;
}
```

2) 3 分
在方法 1 的基础上加了线程锁

```
+ (id)sharedInstance {
    static testClass *sharedInstance = nil;
    @synchronized(self) {
        if (!sharedInstance) {
            sharedInstance = [[self alloc] init];
        }
```

```
    }
    return sharedInstance;
}
```

3) 5 分

```
+ (id)sharedInstance {
    static testClass *sharedInstance = nil;
    static dispatch_once_t once;
    dispatch_once(&once, ^{
        sharedInstance = [[self alloc] init];
    });
    return sharedInstance;
}
```

4. 如何解决 TableView 卡的问题.

1. 使用不透明视图;(2 分)

不透明的视图可以极大地提高渲染的速度。因此如非必要，可以将 table cell 及其子视图的 opaque 属性设为 YES（默认值）。

其中的特例包括背景色，它的 alpha 值应该为 1（例如不要使用 clearColor）；图像的 alpha 值也应该为 1，或者在画图时设为不透明。

2. 图片异步加载; (2 分)
3. 不要重复创建不必要的 table cell(cell 的重用机制); (2 分)

(下面 4 点 总共 4 分)

4. 减少视图数目;
5. 不要做多余的绘制工作;
6. 不要阻塞主线程;
7. 预渲染图像.

http://blog.sina.com.cn/s/blog_b638dc890101ep3x.html

5. SDWebImage 的原理. (根据自己的理解酌情给分)

1. 入口 setImageWithURL:placeholderImage:options: 会先把 placeholderImage 显示，然后 SDWebImageManager 根据 URL 开始处理图片。
2. 进入 SDWebImageManager-downloadWithURL:delegate:options:userInfo:，交给 SDImageCache 从缓存查找图片是否已经下载 queryDiskCacheForKey:delegate:userInfo:.
3. 先从内存图片缓存查找是否有图片，如果内存中已经有图片缓存，SDImageCacheDelegate 回调 imageCache:didFindImage:forKey:userInfo: 到 SDWebImageManager。
4. SDWebImageManagerDelegate 回调 webImageManager:didFinishWithImage: 到 UIImageView+WebCache 等前端展示图片。
5. 如果内存缓存中没有，生成 NSInvocationOperation 添加到队列开始从硬盘查找图片是否已经缓存。
6. 根据 URLKey 在硬盘缓存目录下尝试读取图片文件。这一步是在 NSOperation 进行的操作，所以回主线程进行结果回调 notifyDelegate:。

7. 如果上一操作从硬盘读取到了图片，将图片添加到内存缓存中（如果空闲内存过小，会先清空内存缓存）。SDImageCacheDelegate 回调 imageCache:didFindImage:forKey:userInfo:。进而回调展示图片。
8. 如果从硬盘缓存目录读取不到图片，说明所有缓存都不存在该图片，需要下载图片，回调 imageCache:didNotFindImageForKey:userInfo:。
9. 共享或重新生成一个下载器 SDWebImageDownloader 开始下载图片。
10. 图片下载由 NSURLConnection 来做，实现相关 delegate 来判断图片下载中、下载完成和下载失败。
11. connection:didReceiveData: 中利用 ImageIO 做了按图片下载进度加载效果。
12. connectionDidFinishLoading: 数据下载完成后交给 SDWebImageDecoder 做图片解码处理。
13. 图片解码处理在一个 NSOperationQueue 完成，不会拖慢主线程 UI。如果有需要对下载的图片进行二次处理，最好也在这里完成，效率会好很多。
14. 在主线程 notifyDelegateOnMainThreadWithInfo: 宣告解码完成，imageDecoder:didFinishDecodingImage:userInfo: 回调给 SDWebImageDownloader。
15. imageDownloader:didFinishWithImage: 回调给 SDWebImageManager 告知图片下载完成。
16. 通知所有的 downloadDelegates 下载完成，回调给需要的地方展示图片。
17. 将图片保存到 SDImageCache 中，内存缓存和硬盘缓存同时保存。写文件到硬盘也在以单独 NSInvocationOperation 完成，避免拖慢主线程。
18. SDImageCache 在初始化的时候会注册一些消息通知，在内存警告或退到后台的时候清理内存图片缓存，应用结束的时候清理过期图片。
19. SDWI 也提供了 UIButton+WebCache 和 MKAnnotationView+WebCache，方便使用。
20. SDWebImagePrefetcher 可以预先下载图片，方便后续使用

6. 将一个函数在主线程执行的 4 种方法.(10 分)

GCD 方法 (3 分)
通过向主线程队列发送一个 block 块，使 block 里的方法可以在主线程中执行。

```
dispatch_async(dispatch_get_main_queue(), ^{
    //需要执行的方法
});
```

NSOperation 方法 (3 分)

```
NSOperationQueue *mainQueue = [NSOperationQueue mainQueue];   //主队列
NSBlockOperation *operation = [NSBlockOperation blockOperationWithBlock:^{
    //需要执行的方法
}];
[mainQueue addOperation:operation];
```

NSThread 方法 (2 分)

```
[self performSelector:@selector(method) onThread:[NSThread mainThread] withObject:nil waitUntilDone:YES modes:nil];
```

[self performSelectorOnMainThread:@selector(method) withObject:nil waitUntilDone:YES];

[[NSThread mainThread] performSelector:@selector(method) withObject:nil];
RunLoop 方法 (3 分)
[[NSRunLoop mainRunLoop] performSelector:@selector(method) withObject:nil];

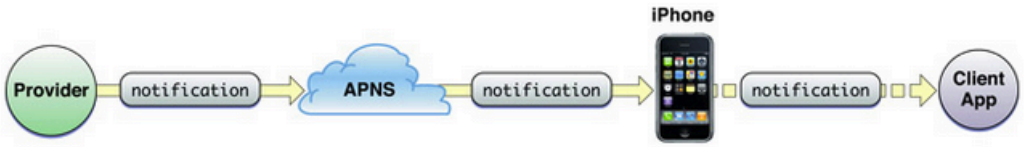
7. 使用 GCD 实现图片的异步加载.(10 分)
能写出加粗部分得 5 分

```
dispatch_async(dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_DEFAULT, 0), ^{
    NSURL * url = [NSURL URLWithString:@"http://avatar.csdn.net/2/C/D/1_totogo2010.jpg"];
    NSData * data = [[NSData alloc]initWithContentsOfURL:url];
    UIImage *image = [[UIImage alloc]initWithData:data];
    if (data != nil) {
        dispatch_async(dispatch_get_main_queue(), ^{
            self.imageView.image = image;
        });
    }
});
```

8. APNs 推送机制是怎样的.(10 分)

APNS的推送机制

首先我们看一下苹果官方给出的对iOS推送机制的解释。如下图



Provider就是我们自己程序的后台服务器，APNS是Apple Push Notification Service的缩写，也就是苹果的推送服务器。

上图可以分为三个阶段：

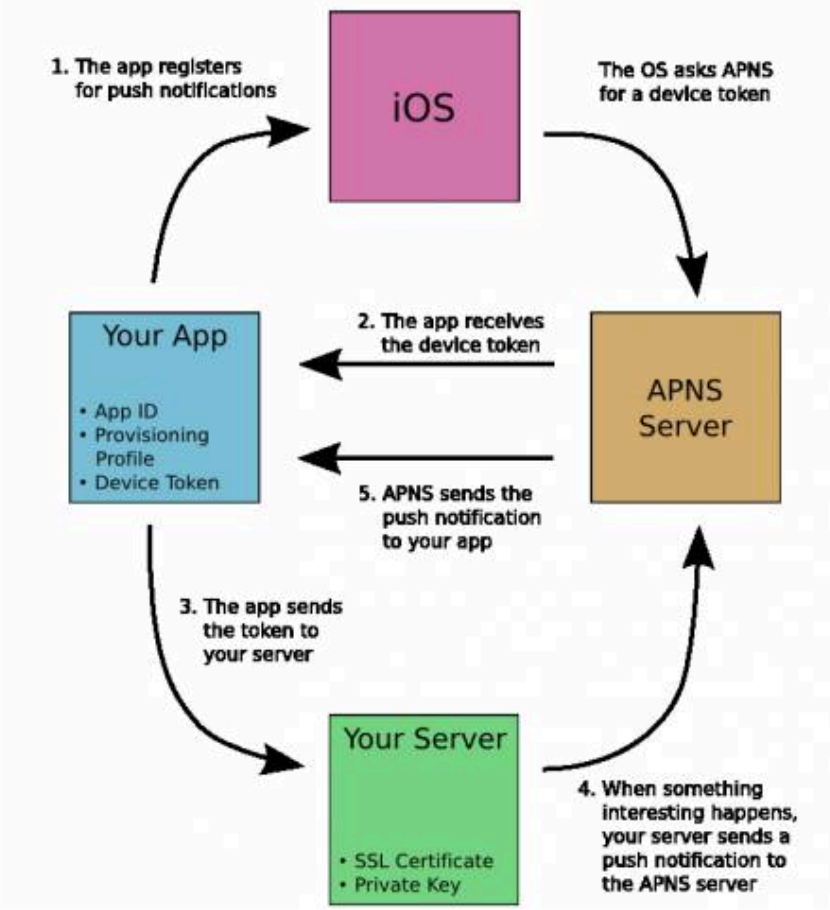
第一阶段：应用程序的服务器端把要发送的消息、目的iPhone的标识打包，发给APNS。

第二阶段：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，并把消息发送到iPhone。

第三阶段：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。

APNS推送通知的详细工作流程

下面这张图是说明APNS推送通知的详细工作流程：



根据图片我们可以概括一下：

1、应用程序注册APNS消息推送。

2、iOS从APNS Server获取devicetoken，应用程序接收device token。

3、应用程序将device token发送给程序的PUSH服务端程序。

4、服务端程序向APNS服务发送消息。

5、APNS服务将消息发送给iPhone应用程序。

9. 页面传值有哪些方式.(10 分)

1.正向传值 VC. (2 分)

2.通知传值; (2 分)

3.代理传值; (2 分)

4.block 传值;(2 分)

5.数据的本地存储 也能实现传值的目的; (2 分)

10. 本地存储有哪些方式.各自有什么特点.(10 分)

1. NSUserDefaults;
2. plist;
3.归档;
4.数据库;

数据库方面可以扩展继续问问题.FMDB/CoreData/SQLite3
数据库:http://blog.csdn.net/zhuming3834/article/details/51111623

11. 解释下列输出结果(10 分)

```
char str1[] = "abc";
char str2[] = "abc";
const char str3[] = "abc";
const char str4[] = "abc";
const char *str5 = "abc";
const char *str6 = "abc";
char *str7 = "abc";
char *str8 = "abc";
cout << ( str1 == str2 ) << endl;
cout << ( str3 == str4 ) << endl;
cout << ( str5 == str6 ) << endl;
cout << ( str7 == str8 ) << endl;
```

结果是：0 0 1 1 (4 分)
解答：str1,str2,str3,str4 是数组变量，它们有各自的内存空间；
而 str5,str6,str7,str8 是指针，它们指向相同的常量区域.(6 分)

12. 写一个“标准”宏 MIN，这个宏输入两个参数并返回较小的一个.(10 分)

```
#define MIN(A,B) ((A) <= (B) ? (A) : (B))
```

13. 请写出下列代码的输出内容.(10 分)

```
#include
int main(void)
{
    int a,b,c,d;
    a=10;
    b=a++;        (3 分)
    c=++a;        (3 分)
   d=10*a++;     (4 分)
    printf("b，c，d：%d，%d，%d"，b，c，d）;
    return 0;
  }
```

答：10，12，120

14. 写一个冒泡排序算法(10 分)

```
- (void)sort:(NSMutableArray *)arr
```

```
{
    for (int i = 0; i < arr.count; i++) {
        for (int j = 0; j < arr.count - i - 1;j++) {
            if ([arr[j+1]integerValue] < [arr[j] integerValue]) {
                int temp = [arr[j] integerValue];
                arr[j] = arr[j + 1];
                arr[j + 1] = [NSNumber numberWithInt:temp];
            }
        }
    }
    NSLog(@"冒泡排序后：%@",arr);
}
```

15. 写一个选择排序算法.(10 分)

```
- (void)sort:(NSMutableArray *)arr
{
    for (int i = 0; i < arr.count; i ++) {
        for (int j = i + 1; j < arr.count; j ++) {
            if ([arr[i] integerValue] > [arr[j] integerValue]) {
                int temp = [arr[i] integerValue];
                arr[i] = arr[j];
                arr[j] = [NSNumber numberWithInt:temp];
            }
        }
    }
    NSLog(@"选择排序后：%@",arr);
}
```