完成代码，判断一个整数是否是奇数：

public boolean isOdd(int i)

相信相当数量的人都已经在准备吐槽了，只要看过《编程珠玑》的人都知道这道题的答案和其中极为简单的道理。不过别着急骂街，不管你信不信，这道笔试题我拿到的答案好多都长这样：

public boolean isOdd(int i) {  
    if (i % 2 == 1) {  
        System.out.println("是奇数");  
    } else {  
        System.out.println("是偶数");  
    }  
}

然后编译一下，发现错误了，挠挠头，顶多改成这样：

public boolean isOdd(int i) {  
    if (i % 2 == 1) {  
        return true;  
    } else {  
        return false;  
    }  
}

好吧，我承认我在筛选简历的能力可能有一些问题，不过不管你信不信，好多大厂工作了几年的程序员，都会写出如上风格的代码。

于是我继续进行引导：

我：“这个函数的定义要求返回一个什么类型的值？”

候选人看了看题干：“布尔类型。”

我：“那么，你if后面的括号里面的表达式的值是一个什么类型的？”

引导到这一步的时候，依然有高达两成的候选人选择了放弃，表示他们不知道。好吧，我真的不知道你们来面试这个职位的信心何在。不过大部分人想了想，还会回答出正确答案：

候选人：“也是布尔类型。”

我：“然后呢？”

有少量候选人虽然没说出来，但是我能看出来他们觉得这只是一个巧合，并不知道怎么进行下一步。不过，大多数人想了想之后，还是会优化成如下代码：

public boolean isOdd(int i) {  
    return i % 2 == 1;  
}

终于过了第一关了，进行第二关的引导：

我：“那我传进来一个-1呢？”

将近一半的人在想了想之后会嘴硬地表示他们从小被教导只有自然数才有奇数偶数之分，负数没有奇偶这一说。剩余的人接受了这个设定，想了一会儿，改成这样：

public boolean isOdd(int i) {  
    return i % 2 == 1 || i % 2 == -1;  
}

并且在提示之后优化成这样：

public boolean isOdd(int i) {  
    return i % 2 != 0;  
}

好吧，这是迄今为止第一个能通过编译且完全满足了需求的代码实现了。说实话，一开始就写成这样的人，如果没有其他什么明显的缺点的话，我这里基本就能通过了。我承认我的要求比较低，但是来面试的人能直接写出这样的真的不太多，粗略地估计的话，大概占一到两成吧。

但是这里还没完呢，还有最重要的第三关呢：

我：“有更好的办法吗？”

候选人：“？”

我：“我觉得取模操作比较慢，有更快的解决方案吗？”

除了少数人能自己想想就想出来了之外，绝大部分（毫不夸张）候选人表示没有或者不知道，于是进行下一步提示：

我：“奇数和偶数转换成二进制有什么区别？”

相当一部分候选人表示自己不懂什么叫二进制和位运算，有的还表示java不是c语言，不用研究这些，就跟很多评论会吐槽我在装逼一样。少部分候选人想了想，会怯怯地回答。

候选人：“奇数最后一位是1，偶数最后一位是0。”

我：“然后呢？”

这里很奇怪的点是，大部分能聊到这里来的候选人会想起来移位操作，我真的不知道是为什么，虽然这道题确实可以有这种操作：

public boolean isOdd(int i) {  
    return i >> 1 << 1 != i;  
}

但是这根本不是重点好吧！！！

总之，无论如何，能在第三关的各种引导之后，能写出下面这个结果来的人，真的不多。能一开始没有任何引导的就写出来的人，至今只见过两个，一个我去哪儿都带着，一个拒了我的offer。

public boolean isOdd(int i) {  
    return (i & 1) == 1;  
}

别以为这就完了！终极boss来了：

我：“这样是不是比上面取模运算要快？”

候选人：“那当然了，位运算肯定快啊。”

我：“但是我们实际代码测试过，发现上面的按位与操作和取模操作，实际运行的时间是差不多的，为什么呢？”