Task1：

Checker 的判断机制是：

1.首先检测字符串的长度是否是 12，如果不是，返回 false。

2.将字符串分成两段，前 8 个为一段 s1，后 4 个为一段 s2。分别传入 checkStr1 和 checkStr2 这两个函数中，如果两个函数都返回 true，则 Checker 返回 true。

1.checkStr1 函数：

将 s1 的 8 个字符顺序取出，依次将字符的 ASCII 值和索引的 11 倍进行异或操作，并和 Checker 中的 secret 数组进行比较，如果都一致就返回 true。

Secret 值	索引值*11	ASCII 值	字符
0x70	0x0	0x70	p
0x64	0xB	0x6F	o
0x64	0x16	0x72	r
0x44	0x21	0x65	e
0x1F	0x2C	0x33	3
0x5	0x37	0x32	2
0x72	0x42	0x30	0
0x78	0x4D	0x35	5

所以前 8 个字符是

pore3205

2.checkStr2 函数：

后面的 4 个字符，首先调用函数将其转化为一个数字 a：

（1）如果 a<1000 返回 false

（2）如果 a 整除 16 或者 27 返回 true

（3）如果 a 不整除 16 或者 27，但个位是 8，也返回 true

（4）其他返回 false

所以有多种答案，如 1458、1008 等

在 android 虚拟机运行：

```
alvikvm -cp CheckBox.dex CheckBox task1
input: pore32051008
task 1: true
root@generic_x86:/data/local/tmp # dalvikvm -cp CheckBox.dex CheckBox task1
input: pore32051458
task 1: true
```

用 java 运行：

```
d:\文件\逆向工程\pore_19300240012\lab4\tasks>cd "d:\文件\逆向工程\pore_19300240012\lab4\tasks\" && javac CheckBox.java && java
 CheckBox task1
input: pore32051008
task 1: true

d:\文件\逆向工程\pore_19300240012\lab4\tasks>cd "d:\文件\逆向工程\pore_19300240012\lab4\tasks\" && javac CheckBox.java && java
 CheckBox task1
input: pore32051458
task 1: true
```

Task2：

Encoder 中共有 5 个函数：

1. convertHexToString：

将字符串两个为一组，转化为 16 进制，和 0xFF 进行异或，将所得的数字，转化为字符，最后存回去。

2. convertStringToHex：

是 convertHexToString 的逆函数，一个字符串先后调用这两了函数保持不变。

将字符串中的每一个字符读出来，取其整数值和 0xFF 取异或，转化为 16 进制的字符串，放回去。

3. getSalt：

创建 6 个 8 位的随机数字，存在 byte[]中，用 random 生成

4. encode：

（1）首先判断数字是否为 11 位，如果不是输出 input error!返回""

（2）调用 getSalt 方法，获得 byte 数组

（3）调用 convertStringToHex 对输入 s+"a"进行转化 得到 hexStr

（4）将 hexStr（共 24 个字符）每 4 个字符一组，取对应 salt，将 salt 转化为 16 进制先存入 stringbuffer，利用 salt%4，前 salt%4 个字符和剩下的字符交换位置后，再放入 stringbuffer。

不断循环处理完 24 个字符，得到 30 个字符。

5. decode：

是 encode 的逆函数。

（1）输入 30 个字符，5 个为一组，依次取出字符，1 一个字符为 x，则 4-x%4 为恢复字符串顺序要交换的位置，将后面 4 个字符串恢复顺序后加入 stringbuffer。

（2）把恢复的字符调用 convertHexToString

（3）把得到的结果删去最后添加的“a”，返回。

在 android 虚拟机中运行：

```
alvikvm -cp CheckBox.dex CheckBox task2
input: 19300240012
encode: 8cec6ccccf3dcfc4cbcfdfcec69ecd
decode: 19300240012
root@generic_x86:/data/local/tmp # dalvikvm -cp CheckBox.dex CheckBox task2
input: 19300240012
encode: 8cec65ccfcccfcd9bcfcdfcec9d9ec
decode: 19300240012
root@generic_x86:/data/local/tmp # dalvikvm -cp CheckBox.dex CheckBox task2
input: 19300240012
encode: ec6ce9ccfc0cfcd2cfcbececf9d9ec
decode: 19300240012
```

用 java 运行：

```
d:\文件\逆向工程\pore_19300240012\lab4\tasks>cd "d:\文件\逆向工程\pore_19300240012\lab4\tasks\" && javac Check
Box.java && java CheckBox task2
input: 19300240012
encode: b6cec9ccfcccfcdbfcbcacecf8cd9e
decode: 19300240012

d:\文件\逆向工程\pore_19300240012\lab4\tasks>cd "d:\文件\逆向工程\pore_19300240012\lab4\tasks\" && javac Check
Box.java && java CheckBox task2
input: 19300240012
encode: 8cec6ccccfccfcd2cfcbbecfc69ecd
decode: 19300240012

d:\文件\逆向工程\pore_19300240012\lab4\tasks>cd "d:\文件\逆向工程\pore_19300240012\lab4\tasks\" && javac Check
Box.java && java CheckBox task2
input: 19300240012
encode: ac6ce3fcccecdcfbfcbcececfccd9e
decode: 19300240012
```