Task1：

第一关是要获得正确的 key，显示出 **Success! Let's play a game.**

```
if(!this.flag.booleanValue()) {
    Boolean v0 = HappyTime.getKey(v4);
    this.flag = v0;
    if(v0.booleanValue()) {
        this.hint.setText("Success! Let\'s play a game.");
        this.array = HappyTime.generateArray(v4);
    }

    return;
}
```

密码是通过 getKey()函数来判断

```
public static Boolean getKey(String arg5) {
    int v0 = arg5.length();
    int v1 = 0;
    Boolean v2 = Boolean.valueOf(false);
    if(v0 != 16) {
        return v2;
    }

    while(v1 < arg5.length()) {
        if((((char)(arg5.charAt(v1) ^ HappyTime.str1.charAt(v1 % HappyTime.str1.length()))))) != HappyTime.res[v1]) {
            return v2;
        }

        ++v1;
    }

    return Boolean.valueOf(true);
}
```

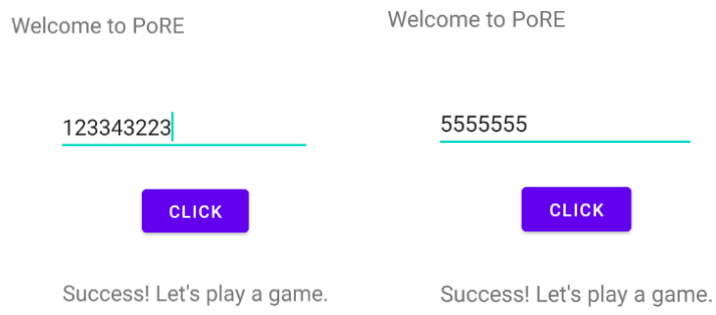
乍一看密码有点复杂，由于可以修改函数，直接在 HappyTime.getKey()后面直接插入代码，把 v0 改为 true。

*#添加三行，将 v0 直接该为 1 的 Boolean 形式*

```
const/4 v0, 0x1

invoke-static {v0}, Ljava/lang/Boolean;->valueOf(Z)Ljava/lang/Boolean;

move-result-object v0

iput-object v0, p0, Lcom/pore/haveagoodtime/MainActivity;->flag:Ljava/lang/Boolean;
```

……

对任意输入均有：




# 但是！

后面发现 task1 的输入直接影响了 task3 的 flag，所以不可以这么暴力，还是要阅读源码。

输入先判断长度是否为 16，然后依次循环和“smali”做异或，循环“smali”字符串，异或的结果和 res 数组比较：

| smali | 16进制 | res | ASCII | 输入 |
|---|---|---|---|---|
| s | 0x73 | 0x23s | 0x50 | P |
| m | 0x6d | 0x22s | 0x4f | O |
| a | 0x61 | 0x33s | 0x52 | R |
| l | 0x6c | 0x29s | 0x45 | E |
| i | 0x69 | 0x19s | 0x70 | p |
| s | 0x73 | 0x1cs | 0x6f | o |
| m | 0x6d | 0x1fs | 0x72 | r |
| a | 0x61 | 0x4s | 0x65 | e |
| l | 0x6c | 0x3cs | 0x50 | P |
| i | 0x69 | 0x6s | 0x6f | o |
| s | 0x73 | 0x21s | 0x52 | R |
| m | 0x6d | 0x28s | 0x45 | E |
| a | 0x61 | 0x11s | 0x70 | p |
| l | 0x6c | 0x23s | 0x4f | O |
| i | 0x69 | 0x1bs | 0x72 | r |
| s | 0x73 | 0x16s | 0x65 | e |

所以 key 是：
POREporePoREpOre

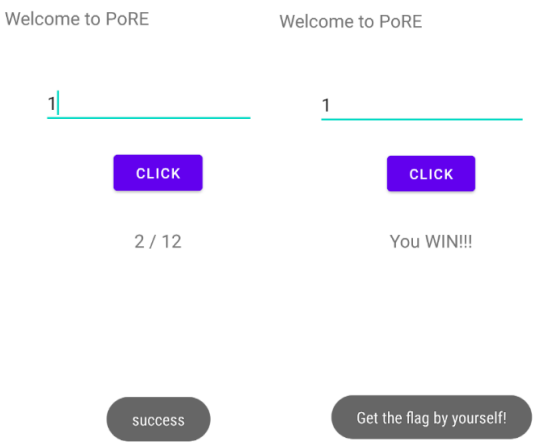
Task2：
第二关是完 12 次游戏，

```
public void playGame(String arg5) {
    int v0 = this.random.nextInt(10000);
    if(arg5.equals(String.valueOf(v0))) {
        Toast.makeText(this.getApplicationContext(), "success", 1).show();
        ++this.times;
        this.array = HappyTime.crypt(this.array, 0, 1);
    }
    else {
        Context v5 = this.getApplicationContext();
        StringBuilder v2 = new StringBuilder();
        v2.append("WRONG, it is ");
        v2.append(v0);
        Toast.makeText(v5, v2.toString(), 1).show();
    }
}
```

要判断随机数，显然不可能赢 12 次，直接在 playGame()中间 if 的条件修改，变成不相等时进入

*#修改，将 eqz 改为 nez*

if-nez p1, :cond_0




一直输入 1，最后很容易就获胜。

Task3：

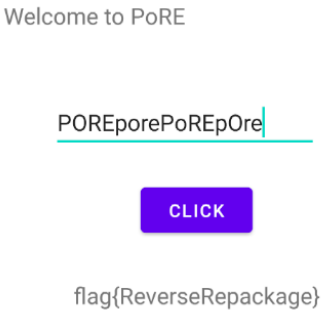
前两个任务，每一个除了必要的任务逻辑函数，还对一个 array 进行了修改，由此推测，就是要获得这个 array，有关这个数组的函数是 native 的，无法直接获得。

发现在 MainActivity 中有 show（）函数从未调用过，而且其中就含有 flag 和 array，猜测是要在我们主动来调用。

```
public void show() {
    this.hint.setText(String.format("flag{%s}", new String(this.array)));
}
```

添加一行：

```
#直接调用 show 函数

invoke-virtual {p0}, Lcom/pore/haveagoodtime/MainActivity;->show()V
```




此时如果 task1：直接暴力跳过，结果如下，全为乱码


所以 flag 为：
ReverseRepackage