# 第 1 题

等价类表：

| 序号 | 有效等价类 | 编号 | 无效等价类 | 编号 |
|---|---|---|---|---|
| 1 | 10 位接种号 | 1 | 接种号<10 位 | 2 |
| | | | 接种号>10 位 | 3 |
| | | | 空 | 4 |
| 2 | 首位大写字母属于{A, B, C} | 5 | 首位不是字母 | 6 |
| | | | 首位是小写字母 | 7 |
| | | | 首位大写但不属于{A, B, C} | 8 |
| 3 | 2~3 位表示接种月份 | 9 | 2~3 位不全是数字 | 10 |
| | | | 2~3 位是数字但全 0 或>12 | 11 |
| 4 | 4~5 位属于{07, 13, 23} | 12 | 4~5 位不是数字 | 13 |
| | | | 4~5 位是数字但不属于{07, 13, 23} | 14 |
| 5 | 后 5 位为数字编号 | 15 | 后 5 位不是数字 | 16 |

覆盖数据（标红的是无效等价类的编号）：

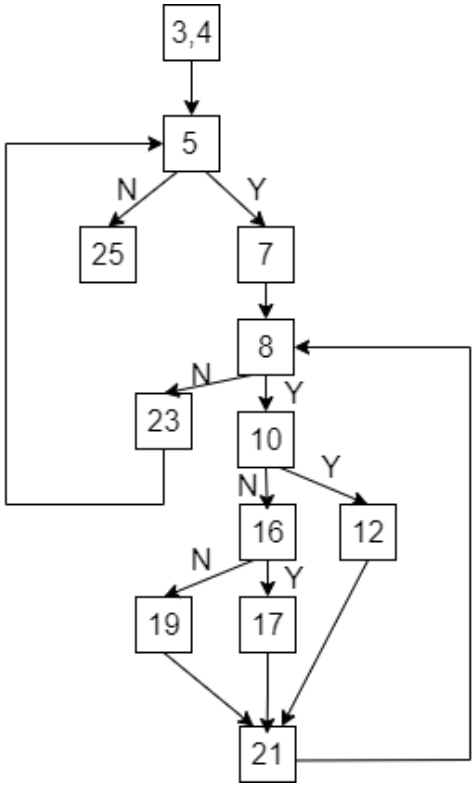
| 序号 | 数据 | 类型 | 覆盖的等价类编号 |
|---|---|---|---|
| 1 | A 11 07 00000 | 有效 | 1,5,9,12,15 |
| 2 | A 11 07 0000 | 无效 | 2,5,9,12 |
| 3 | A 11 07 000000 | 无效 | 3,5,9,12 |
| 4 | | 无效 | 4,5,9,12 |
| 5 | 2 11 07 00000 | 无效 | 1,6,9,12,15 |
| 6 | a 11 07 00000 | 无效 | 1,7,9,12,15 |
| 7 | Z 11 07 00000 | 无效 | 1,8,9,12,15 |
| 8 | A ab 07 00000 | 无效 | 1,5,10,12,15 |
| 9 | A 13 07 00000 | 无效 | 1,5,11,12,15 |
| 10 | A 11 ab 00000 | 无效 | 1,5,9,13,15 |
| 11 | A 11 12 00000 | 无效 | 1,5,9,14,15 |
| 12 | A 11 07 0a000 | 无效 | 1,5,9,12,16 |

# 第 2 题

Q2-1

程序的流图如下所示，其中每个节点表示代码的行号。共有 12 个节点，15 条边。

故环路复杂度： $V(G) = E - N + 2 = 15 - 12 + 2 = 5.$




这是一个求 Longest Common Subsequence 的非递归算法。

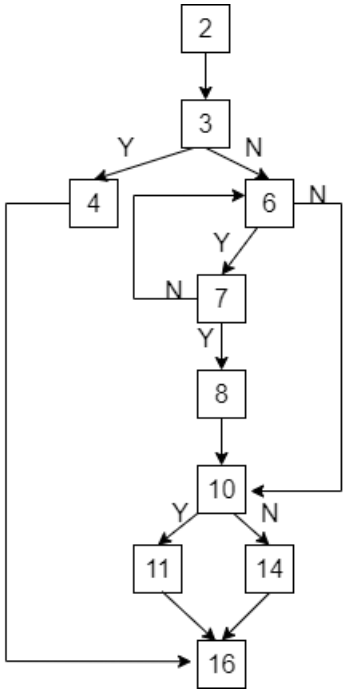
所有独立路径:

| 序号 | 独立路径 | 测试用例 | 输出 |
|---|---|---|---|
| 1 | $(3,4) \to 5 \to 25$ | $lcs(2,1,"a","a")$ | 0 |
| 2 | $(3,4) \to 5 \to 7 \to 8 \to 23 \to 5 \to 25$ | $lcs(1,2,"a","a")$ | 0 |
| 3 | $(3,4) \to 5 \to 7 \to 8 \to 10 \to 16 \to 19 \to 21 \to 8 \to 23 \to 5 \to 25$ | $lcs(1,1,"a","b")$ | 0 |
| 4 | $(3,4) \to 5 \to 7 \to 8 \to 10 \to 16 \to 17 \to 21 \to 8 \to 23 \to 5 \to 25$ | $lcs(1,1,"abc","acb")$ | 2 |
| 5 | $(3,4) \to 5 \to 7 \to 8 \to 10 \to 12 \to 21 \to 8 \to 23 \to 5 \to 25$ | $lcs(1,1,"ab","a")$ | 1 |

Q2-2

程序的流图如下所示，其中每个节点表示代码的行号。共有 10 个节点，13 条边。

故环路复杂度： $V(G) = E - N + 2 = 13 - 10 + 2 = 5.$




所有独立路径:

| 序号 | 独立路径 | 测试用例 | 输出 |
|---|---|---|---|
| 1 | $2 \to 3 \to 4 \to 16$ | $n = 1$ | 不是素数 |
| 2 | $2 \to 3 \to 6 \to 7 \to 6 \to 10 \to 11 \to 16$ | 无 | 不存在这条路径 |
| 3 | $2 \to 3 \to 6 \to 7 \to 6 \to 10 \to 14 \to 16$ | $n = 3$ | 是素数 |
| 4 | $2 \to 3 \to 6 \to 7 \to 8 \to 10 \to 11 \to 16$ | $n = 4$ | 不是素数 |
| 5 | $2 \to 3 \to 6 \to 7 \to 8 \to 10 \to 14 \to 16$ | 无 | 不存在这条路径 |
| 6 | $2 \to 3 \to 6 \to 10 \to 11 \to 16$ | 无 | 不存在这条路径 |
| 7 | $2 \to 3 \to 6 \to 10 \to 14 \to 16$ | $n = 2$ | 是素数 |

# 第 3 题

The bugs found in SonarQube analysis is not code bugs instead its code quality bugs. That means if there are any code quality issues, that will be considered as bug in SonarQube Analysis and clearly its not related to regular bugs which can be found in application functionalities. Bugs found in SonarQube may not harming functionality of the application, but it indicates code quality issues.

由上述文字知，SonarQube 主要检查会影响代码质量的风格（而不一定是语法错误）。

检查出的错误如下:

1. 使用未定义的函数或结构体:

$add(), sub(), MyClass$ 均是还未声明定义的时候就使用了。


调整使用和声明定义的位置，即可改正错误：

```
# define after call
def myALU():
    add = sum
    add()

    def sub():
        pass
    sub()

    class MyClass:
        pass
    MyClass()
```

2. 调用函数时参数个数与声明时的不匹配：

$print\_2\_args()$ 声明时参数不超过 2 个，第 22 行调用时传了 3 个参数，第 23 行调用时没有传参数。


传入正确的参数个数，即可改正错误：

```
# function arguments
param_args = [1, 2, 3]
param_kwargs = {'x': 1, 'y': 2}
```

```
def print_2_args(a, b=1):
    print(a, b)

print_2_args(1)
```

3. 对同个变量的比较操作（如用==或!=比较）：
第 32、35、38 行都对相同的变量（$a, b, a == b$）进行比较，这会导致分支一定执行/不执行，影响代码质量。

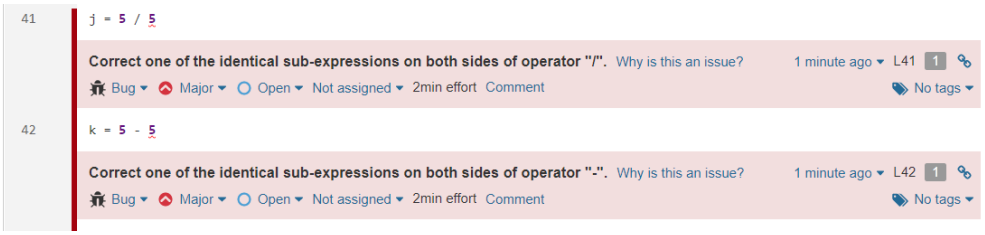

修改比较的条件，即可改正错误：

```
# identical value compare/calculate
if a == b: # Noncompliant
    print("equal!")

if  a != b: # Noncompliant
    print("not equal!")

if  a == b and a == c: # Noncompliant
    print("a equal to b and a equal to c!")
```

4. 对相同的变量/数字进行数值运算：
第 41、42 行对相同的数字 5 进行运算，会得到恒定的值（1 和 0），影响代码质量。




修改运算数，使其不相等，即可改正错误：

```
j = 5 / 3
k = 5 - 4
```

5. 忽略函数的参数，直接对其赋值：

$myFunc()$ 函数的参数 $integers$ 的值还没保存，就直接对其赋值，容易丢失信息。


先保存参数值，再对其进行赋值，即可改正错误：

```
# ignore function's params' values
def myFunc(strings, integers):
    store = integers
    integers = 1  # NonCompliant
```

修改后再运行$sonar-scanner$的结果：