# Answer

**PB22111639 马筱雅**

# 正则表达式

**\w+([-+.]\w+)*@\w+([-.]\w+)*.\w+([-.]\w+)* 正则表达式匹配的字符串的含义是什么?**

- 可以分成 `\w+`，`([-+.]\w+)*`，`@\w+`，`([-.]\w+)*`，`\.\w+`，`([-.]\w+)*`
- 该正则表达式每个 `-+.` 字符后面一定跟有由字母数字或者下划线组成的串，且一定包含有 字符串@字符串.字符串，
- 根据邮箱的规则，可以得到该正则表达式表示电子邮箱

# Flex

1. **第三条规则**
   scanner从输入字符串开头开始扫描，则 `+` 与第一条规则和第三条规则匹配，接着匹配 `=`，和第三条规则匹配，选择匹配最多的规则，即第三条规则
2. **第一条规则**
   输入字符串 `ABC` 与第一条规则的 `ABC` 完全匹配，与第二条规则也匹配，触发位于前面的规则，所以第一条规则会被触发
3. **第一条规则**
   `ABC` 与第一条规则和第二条规则均完全匹配，当匹配字符串的长度相同时，位于前的规则被触发，故第一条规则被触发。

# Bison

1.**上述计算器例子的文法中存在左递归，为什么 bison 可以处理?**

在计算器例子中，采用自下而上的归约方式构建语法树, 并且bison采用LALR文法，所以可以处理左递归

2. **能否修改计算器例子的文法，使得它支持除数 0 规避功能?**

将

```
term MULOP factor
{
    switch ($2) {
    case '*': $$ = $1 * $3; break;
    case '/': $$ = $1 / $3;break; // 这里会出什么问题？
    }
}
```

修改为

```
term MULOP factor{
    switch ($2) {
    case '*': $$ = $1 * $3; break;
    case '/': if($3 == 0){printf("error 0\n"); return 0;}
            else {$$ = $1 / $3;} break; // 这里会出什么问题？
    }
}
```