第二题是

定义了3个计算 ： ！ & |

其中 ！的优先级最高，&次之，|最低。还有括号，高于所有优先级

！ 取非运算 （0变成1,1变成0）

& 与运算 （0 & 1 = 0， 1 & 0 = 0， 0 & 0 = 0， 1 & 1 = 1）

| 或运算 （0 | 0 = 0, 1 | 0 = 1, 0 | 1 = 1, 1 | 1 = 1） （试卷上的原题目写错了，或运算应该是这么定义的，意思是一样的）

现在输入一个表达式如：
"!(1&0)|0&1"
运算结果为1

我的做法是：
其实和算术表达式类似，只是各个运算符的优先级的定义，还有所需要的操作数个数不通而已

1、将原表达式转换为逆波兰表达式（后缀表达式）
如"!(1&0)|0&1" 转换为了 "10&!0|&1"
2、再根据得到的逆波兰表达式计算整个表达式的结果

```
#include<iostream>
#include<stack>
#include<queue>
#include<map>
using namespace std;

map<char, int> priority;

//原表达式转换为逆波兰表达式
queue<char> getPostExp(string str) {
    stack<char> op;
    queue<char> res;
    int n = str.size();
    for (int i = 0; i < n; i++) {
        char c = str[i];
        if (c >= '0' && c <= '9') {
            res.push(c);
        }
        else {
            if (c == '(') {
                op.push(c);
                continue;
            }
            if (c != ')') {
                if (op.empty()) {
                    op.push(c);
                    continue;
                }
                else {
                    int order = priority[c];
```

```
                    while (!op.empty() && priority[op.top()] >= order) {
                        res.push(op.top());
                        op.pop();
                    }
                    op.push(c);
                    continue;
                }
            }
            if (c == ')') {
                while (!op.empty() && op.top() != '(') {
                    res.push(op.top());
                    op.pop();
                }
                if (op.top() == '(') op.pop();
            }
        }
    }
    while (!op.empty()) {
        res.push(op.top());
        op.pop();
    }
    return res;
}

//计算逆波兰表达式
int getResult(queue<char> res) {
    stack<int> nums;
    int result = 0;
    int num1, num2;
    while (!res.empty()) {
        char c = res.front();
        res.pop();
        if (!(c >= '0' && c <= '9')) {
            if (c != '!') {
                num1 = nums.top();
                nums.pop();
                num2 = nums.top();
                nums.pop();
                if (c == '&') {
                    result = num1 & num2;
                    nums.push(result);
                }
                if (c == '|') {
                    result = num1 | num2;
                    nums.push(result);
                }
            }
            else {
                num1 = nums.top();
                nums.pop();
                result = !num1;
                nums.push(result);
            }
        }
        else {
            nums.push(c - '0');
        }
    }
```

```
    return nums.top();
}

int main() {
    priority['!'] = 3;
    priority['&'] = 2;
    priority['|'] = 1;
    //string str = "!(1&0)&0|0"; //The answer is 0
    //string str = "!(1&0)|0&1"; //The answer is 1;
    //string str = "(1&1|0)|(1&0)&(0|0)"; //The answer is 1
    string str;
    while (cin >> str) {
        queue<char> res = getPostExp(str);
        /*while (!res.empty()) {
            cout << res.front() << " ";
            res.pop();
        }*/
        int result = getResult(res);
        cout << result << endl;
    }
    return 0;
}
```