

```

1 #ifndef OPERAND_H
2 #define OPERAND_H
3
4 class operand{
5 public:
6     int p; char o;
7     operand(){}
8     operand(const int pre, const char opd):
9         p(pre), o(opd){}
10    operand& operator()(const int pre, const char opd){
11        p=pre; o=opd;
12        return *this;
13    }
14    bool operator<(const operand& opd){
15        return p<opd.p;
16    }
17    bool operator>(const operand& opd){
18        return p>opd.p;
19    }
20    bool operator<=(const operand& opd){
21        return p<=opd.p;
22    }
23    bool operator>=(const operand& opd){
24        return p>=opd.p;
25    }
26    bool operator&&(const operand& opd){
27        return p && opd.p;
28    }
29    bool operator&&(const int& opd){
30        return p && opd;
31    }
32    bool operator||(const int& opd){
33        return p || opd;
34    }
35    bool operator||(const operand& opd){
36        return p||opd.p;
37    }
38    bool operator!(){
39        return !p;
40    }
41    bool operator==(const operand& opd){
42        return p==opd.p;
43    }
44    bool operator!=(const operand& opd){
45        return p!=opd.p;
46    }
47    bool operator==(const int& m){
48        return p==m;
49    }
50    bool operator!=(const int& m){
51        return p!=m;
52    }
53    bool operator>(const int& m){
54        return p>m;
55    }
56    bool operator<(const int& m){
57        return p<m;
58    }
59    bool operator<=(const int& m){
60        return p<=m;
61    }
62    bool operator>=(const int& m){
63        return p>=m;
64    }
65    operand& operator=(const operand& opd){
66        p=opd.p;
67        o=opd.o;
68        return *this;

```

```
69     }  
70  
71 };  
72 #endif
```