```java
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.ArrayDeque;
import java.util.List;
import java.util.ArrayList;
import java.util.Queue;
import java.util.SortedSet;
import java.util.TreeSet;
/**
 *
 * @author tym11
 */
public class samples {
    static class Sample implements Comparable<Sample> {
        final int t;
        final int v;
        Sample(int t, int v) {
            this.t = t;
            this.v = v;
        }
        @Override
        public String toString() {
            return "t=" + t + " v=" + v;
        }

        boolean isInTimeInterval(Sample fromSample, int L) {
            return t >= fromSample.t - L;
        }

        @Override
        public int compareTo(Sample o) {
            return Integer.compare(v, o.v);
        }

        @Override public boolean equals(Object o) {
            return o instanceof Sample && ((Sample)o).compareTo(this) == 0;
        }
    }

    private enum OP {
        gt, lt;
    }

    private enum AG {
        min, max, avg;
    }

```

```
49:     static class Rule {
50:         final OP op;
51:         final AG ag;
52:         final int L;
53:
54:         Rule(OP op, AG ag, int L) {
55:             this.op = op;
56:             this.ag = ag;
57:             this.L = L;
58:         }
59:
60:         @Override
61:         public String toString() {
62:             return op.name() + " " + ag.name() + " " + L;
63:         }
64:     }
65:
66:     /**
67:      * @param args the command line arguments
68:      */
69:     public static void main(String[] args) throws IOException {
70:         BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
71:
72:         for(;;) {
73:             String s = in.readLine();
74:             if (s == null)
75:                 break;
76:
77:             int n = Integer.valueOf(s);
78:             List<Sample> samples = new ArrayList<>(n);
79:             for (int i = 0; i < n; ++i) {
80:                 String[] f = in.readLine().split(" ");
81:                 samples.add(new Sample(Integer.valueOf(f[0]), Integer.valueOf(f[1])));
82:             }
83:
84:             int nRules = Integer.valueOf(in.readLine());
85:             List<Rule> rules = new ArrayList<>(nRules);
86:             for (int i = 0; i < nRules; ++i) {
87:                 String[] f = in.readLine().split(" ");
88:                 rules.add(new Rule(OP.valueOf(f[0]), AG.valueOf(f[1]), Integer.valueOf(f[2])));
89:             }
90:
91:             solve(samples, rules);
92:         }
93:     }
94:
95:     private static void solve(List<Sample> samples, List<Rule> rules) {
96:         for (Rule r: rules) {
```

```
97:             int satisfied = 0;
98:             long[] ev = new long[samples.size()]; // evaluated window for j < i
99:             int[] ws = new int[samples.size()]; // window size
100:
101:            SortedSet<Sample> ss = new TreeSet<>();
102:            Queue<Sample> q = new ArrayDeque<>();
103:            if (r.ag == AG.avg) {
104:                long sum = 0;
105:                for (int i = 0; i < samples.size(); ++i) {
106:                    Sample si = samples.get(i);
107:
108:                    // odstran zacatek fronty mimo okno vzhledem k sample[i]
109:                    Sample rem;
110:                    while ((rem = q.peek()) != null && !rem.isInTimeInterval(si, r.L)) {
111:                        sum -= q.remove().v; // update souctu fronty
112:                    }
113:
114:                    // zapis okno do [i]
115:                    ev[i] = sum; // soucet vsech hodnot ve fronte
116:                    ws[i] = q.size(); // velikost fronty
117:
118:                    q.add(si); // pridej sample[i] do fronty
119:                    sum += si.v; // update souctu fronty
120:                }
121:                for (int i = 0; i < samples.size(); ++i) {
122:                    Sample si = samples.get(i);
123:                    if (ws[i] > 0) {
124:                        if (r.op == OP.lt && si.v < ev[i]/ws[i]) {
125:                            satisfied++;
126:                        } else if (r.op == OP.gt && si.v > ev[i]/ws[i]) {
127:                            satisfied++;
128:                        }
129:                    }
130:                }
131:            } else if (r.ag == AG.min) {
132:                for (int i = 0; i < samples.size(); ++i) {
133:                    Sample si = samples.get(i);
134:
135:                    // odstran zacatek fronty mimo okno vzhledem k sample[i]
136:                    Sample rem;
137:                    while ((rem = q.peek()) != null && !rem.isInTimeInterval(si, r.L)) {
138:                        ss.remove(q.remove());
139:                    }
140:
141:                    ws[i] = q.size(); // velikost fronty
142:                    ev[i] = ss.isEmpty() ? 0 : ss.first().v; // min vsech hodnot ve fronte
143:
144:                    q.add(si); // pridej sample[i] do fronty
```

```
145:                     ss.add(si);
146:                 }
147:                 for (int i = 0; i < samples.size(); ++i) {
148:                     Sample si = samples.get(i);
149:                     if (ws[i] > 0) {
150:                         if (r.op == OP.lt && si.v < ev[i]) {
151:                             satisfied++;
152:                         } else if (r.op == OP.gt && si.v > ev[i]) {
153:                             satisfied++;
154:                         }
155:                     }
156:                 }
157:             } else if (r.ag == AG.max) {
158:                 for (int i = 0; i < samples.size(); ++i) {
159:                     Sample si = samples.get(i);
160:
161:                     // odstran zacatek fronty mimo okno vzhledem k sample[i]
162:                     Sample rem;
163:                     while ((rem = q.peek()) != null && !rem.isInTimeInterval(si, r.L)) {
164:                         ss.remove(q.remove());
165:                     }
166:
167:                     ws[i] = q.size(); // velikost fronty
168:                     ev[i] = ss.isEmpty() ? 0 : ss.last().v; // max vsech hodnot ve fronte
169:
170:                     q.add(si); // pridej sample[i] do fronty
171:                     ss.add(si);
172:                 }
173:                 for (int i = 0; i < samples.size(); ++i) {
174:                     Sample si = samples.get(i);
175:                     if (ws[i] > 0) {
176:                         if (r.op == OP.lt && si.v < ev[i]) {
177:                             satisfied++;
178:                         } else if (r.op == OP.gt && si.v > ev[i]) {
179:                             satisfied++;
180:                         }
181:                     }
182:                 }
183:             }
184:
185:             System.out.println(satisfied);
186:         }
187:     }
188: }
```