爱奇艺累计有2000万会员，为了提高会员权益以及减少运营成本，我们需要对用户分等级差异化运营，为此我们制定了用户成长体系，按成长值（最小为0）划分为不同等级。成长值计算公式为：  
会员成长值＝每天成长值＋任务成长值  
现在我们输入一组数据，表示用户的成长值计算规则，比如某个用户的每天成长值规则1 1 5 10，第一列1表示每日成长值规则，第二列1表示该条规则的生效开始时间，第三列5表示该条规则的生效截至时间，第四列10表示该条规则的每天成长值，则用户初始值为0，第1天到第5天，每天成长值10点，则第5天成长值为50；另外任务成长值规则，比如2 3 4，第一列2表示该规则为任务成长值，第二列3表示第三天做任务，第三列4表示该天做任务得到成长值4。现在输入一组数据，每行一条成长规则，每日成长规则生效时间重合时以成长数值最大的为准，每日成长值是每天0点更新，任务成长值是0点以后，要求计算成长值规则对应最后一天成长值。  
输入  
输入数据有多行，第一列为1时，该行会有4个数值，第一列为2时，该列会有3个数值  
输出  
对于每个测试实例，初始成长值都为0，计算成长值规则最后一天的用户成长值。  
  
  
样例输入  
1 1 5 10  
2 3 4  
1 4 6 －5  
样例输出  
49

解决该问题可以用最直接的思路，首先将每条规则存下来，然后从最小天数开始到最大天数依次得到下一个要计算的区间，例如样例中，首先是1-5，每天成长值为10（因为第一条规则第五天的成长值大于第三条，所以第五天的成长值按照第一条规则来计算），下一条规则是5-6，每天成长值为 -5，最后再加上任务的成长值可以得到最总结果为49.

以上思路可以解决问题，但是总体解决思路比较乱，需要考虑的情况有很多，因为有可能很多条规则都有重复，这时候需要遍历这些规则找到最合适下一条规则。

还有另一种思路，即将成长区间转化为每天的成长，化整为零，这样思路就简单了很多。具体操作如下：

首先将每条规则都读取进来，并且在读取的时候就确定开始时间与结束时间，然后声明一个数组，大小为总的天数，数组中每个元素代表该天的成长值。然后再遍历每条规则，将该规则的成长值写入到该数组中，当然，每天的成长值都取最大的那个成长值。最后，将该数组求和即可。

import java.util.Scanner;  
  
public class mayi {  
    private static long solve(int[] arrays, int[] arraysB,int l) {  
        long ans = 0;  
        for (int i = 0; i <= l; i++) {  
            ans += arrays[i];  
            ans += arraysB[i];  
        }  
        return ans;  
    }  
    public static void main(String[] arg) {  
        Scanner in = new Scanner(System.in);  
        int[] arrays = new int[1002];  
        int[] arraysB = new int[1002];  
        int l = 0;  
        int count = 0;  
        int num = 0;  
  
        while (in.hasNext()) {  
            //int num = scan.nextInt();  
            String input = in.nextLine();  
            String[] s = input.split(" ");  
  
            if (s.length == 1) {  
                 num = Integer.parseInt(s[0]);  
                //System.out.println(num);  
            }  
            else if (s.length == 4) {  
                int start = Integer.parseInt(s[1]);  
                int end = Integer.parseInt(s[2]);  
                int value = Integer.parseInt(s[3]);  
                for (int i = start; i <= end; i++) {  
                    if (arrays[i] != 0) {  
                        arrays[i] = arrays[i] > value ? arrays[i] : value;  
                    }else {  
                        arrays[i] = value;  
                    }  
                }  
                l = l > end ? l : end;  
            }else {  
                int time = Integer.parseInt(s[1]);  
                int value = Integer.parseInt(s[2]);  
                arraysB[time] = value;  
                l = l > time ? l : time;  
            }  
           count++;  
            if (count == num+1) {  
            System.out.println(solve(arrays,arraysB,l));  
                count=0;  
            }  
        }  
        in.close();  
    }  
  
}