题目描述：
IGMP 协议中， 有一个字段称作最大响应时间（Max Response Time）， HOST 收到查询报文， 解析出 MaxResponseTime 字段后， 需要在 (0, MaxResponseTime](s) 时间内选取随机时间回应一个响应报文，如果在随机时间内收到一个新的查询报文，则会根据两者时间的大小，选取小的一方刷新回应时间。
最大响应时间有如下计算方式：
当 Max Resp Code < 128, Max Resp Time = Max Resp Code;
当 Max Resp Code >= 128,

```
  0 1 2 3   4   5 6 7
+-+-+-+-+-+-+-+-+
|1 |   exp |   mant   |
+-+-+-+-+-+-+-+-+
```

Max Resp Time = (mant | 0x10) << (exp + 3);
注：exp 最大响应时间的高 5~7 位；mant 为最大响应时间的低 4 位。
其中 接收到的 MaxRespCode 最大值为 255， 以上出现所有字段均为无符号数。
现在我们认为 HOST 收到查询报文时， 选取的随机时间必定为最大值。 现给出 HOST 收到查询报文个数 C， HOST 收到该报文的时间 T，以及查询报文的最大响应时间字段值 M ， 请计算出 HOST 发送响应报文的时间。
输入： 第一行为查询报文个数 C，后续每行分别为 HOST 收到报文时间 T，及最大响应字段 M，以空格分割。
输出： HOST 发送响应报文的时间

输入描述：
第一行为查询报文个数 C，后续每行分别为 HOST 收到报文时间 T，及最大响应时间 M，以空格分割。
输出描述：
HOST 发送响应报文的时间。
补充说明：
用例确定只会发送一个响应报文， 不存在计时结束后依然收到查询报文的情况。

示例 1
输入：
3
0 20
1 10
8 20
输出：
11
说明：
收到 3 个报文，
第 0 秒收到第 1 个报文，响应时间为 20 秒，则要到 0+20=20 秒响应；
第 1 秒收到第 2 个报文，响应时间为 10；则要到 1+10=11 秒响应，与第上面的报文的响应

时间比较获得响应时间最小为 11 秒；
第 8 秒收到第 3 个报文，响应时间为 20 秒，则要到 8+20=28 秒响应；与第上面的报文的响应时间比较获得响应时间最小为 11 秒；
最终得到最小响应报文时间为 11 秒
示例 2
输入：
2
0 255
200 60
输出：
260
说明：
收到 2 个报文，
第 0 秒收到第 1 个报文，响应时间为 255 秒，则要到(15 | 0x10) << (7 + 3) = 31744 秒响应；（mant = 15，exp = 7）
第 200 秒收到第 2 个报文，响应时间为 60；则要到 200+60=260 秒响应，与第上面的报文的响应时间比较获得响应时间最小为 260 秒；
最终得到最小响应报文时间为 260 秒

```
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        // 获取报文数量
        int boardCount = scanner.nextInt();
        int minTime = Integer.MAX_VALUE;
        int curRespTime = 0;
        // 循环进行处理
        for (int i = 0; i < boardCount; i++) {
            curRespTime = 0;
            // 获取单个报文进行分割处理
//              String singleBoard = scanner.nextLine();
//              String[] boardComponent = singleBoard.split(" ");
            int recvTime = scanner.nextInt();
            int respTime = scanner.nextInt();
            if (respTime >= 128) {
                // 需要先转换为二进制
                String binaryStr = Integer.toBinaryString(respTime);
                String expStr = binaryStr.substring(1, 4);
                int exp = Integer.parseInt(expStr, 2);
                String mantStr = binaryStr.substring(4);
                int mant = Integer.parseInt(mantStr, 2);
                respTime = (mant | 0x10) << (exp + 3);
            }
```

```
                curRespTime = recvTime + respTime;
                minTime = Math.min(minTime, curRespTime);
            }

            System.out.println(minTime);
        }
    }
```