一、编程题

ACM：开源项目热榜

题目描述：某个开源社区希望将最近热度比较高的开源项目出一个榜单，推荐给社区里面的开发者。对于每个开源项目，开发者可以进行关注(watch)、收藏(star)、fork、提issue、提交合并请求(MR)等。
数据库里面统计了每个开源项目关注、收藏、fork、issue、MR的数量，开源项目的热度根据这5个维度的加权求和进行排序。
$H = W_{watch} \times \#watch + W_{star} \times \#star + W_{fork} \times \#fork + W_{issue} \times \#issue + W_{mr} \times \#mr$
H表示热度值，W_{watch}、W_{star}、W_{fork}、W_{issue}、W_{mr}分别表示5个统计维度的权重，
$\#watch$、$\#star$、$\#fork$、$\#issue$、$\#mr$分别表示5个统计维度的统计值。
榜单按照热度值降序排序，对于热度值相等的，按照项目名字转换为全小写字母后的字典序排序（'a','b','c',...,'x','y','z'）。

输入描述：第一行输入为N，表示开源项目的个数，$0 < N \leq 100$。
第二行输入为权重值列表，一共5个整型值，分别对应关注、收藏、fork、issue、MR的权重，权重取值$0 < W \leq 50$。
第三行开始接下来的N行为开源项目的统计维度，每一行的格式为：
name nr_watch nr_star nr_fork nr_issue nr_mr
其中name为开源项目的名字，由英文字母组成，长度≤ 50，其余5个整型值分别为该开源项目关注、收藏、fork、issue、MR的数量，数量取值$0 < nr \leq 1000$。

输出描述：按照热度降序，输出开源项目的名字，对于热度值相等的，按照项目名字转换为全小写字母后的字典序排序（'a'>'b'>'c'>...>'x'>'y'>'z'）。

补充说明：

示例1

输入：4
8 6 2 8 6
camila 66 70 46 158 80
victoria 94 76 86 189 211
anthony 29 17 83 21 48
emily 53 97 1 19 218

输出：victoria
camila
emily
anthony

说明：排序热度值计算：
camila: 66*8 + 70*6 + 46*2 + 158*8 + 80*6 = 2784
victoria: 94*8 + 76*6 + 86*2 + 189*8 + 211*6 = 4158
anthony: 29*8 + 17*6 + 83*2 + 21*8 + 48*6 = 956
emily: 53*8 + 97*6 + 1*2 + 19*8 + 218*6 = 2468
根据热度值降序，得到结果。

示例2
输入：5
5 6 6 1 2
camila 13 88 46 26 169
grace 64 38 87 23 103
lucas 91 79 98 154 79
leo 29 27 36 43 178
ava 29 27 36 43 178
输出：lucas
grace
camila
ava
leo
说明：排序热度值计算：
camila: 13*5 + 88*6 + 46*6 + 26*1 + 169*2 = 1233
grace: 64*5 + 38*6 + 87*6 + 23*1 + 103*2 = 1299
lucas: 91*5 + 79*6 + 98*6 + 154*1 + 79*2 = 1829
leo: 29*5 + 27*6 + 36*6 + 43*1 + 178*2 = 922
ava: 29*5 + 27*6 + 36*6 + 43*1 + 178*2 = 922
根据热度值降序，对于leo和ava，热度值相等，按照字典序，ava排在leo前面，得到结果。

代码：

```
const rl = require("readline").createInterface({ input: process.stdin });
var iter = rl[Symbol.asyncIterator]();
const readline = async () => (await iter.next()).value;

void async function () {
    // Write your code here
    const count = parseInt(await readline())
    const rate = (await readline()).split(' ').map(p => parseInt(p))
    const project = []
    for (let i = 0; i < count; i++) {
        const line = await readline()
        let tokens = line.split(' ');
        project.push([tokens[0], h(rate, tokens.slice(1, tokens.length))])
    }
    project.sort((x, y) => y[1] - x[1]).forEach(p => {
        console.log(p[0])
    })

}()

function h(r, v) {
    return r.map((rate, i) => rate * v[i]).reduce((x, y) => x + y)
}
```