第四届胖头鱼杯信息学蒟蒻争霸赛

时间： 2024 年 3 月 9 日 14：30 至 18：00，共 3.5 小时。

题目名称	胖头鱼战士	武器展示	角色配队	深境方块
题目类型	传统型	传统型	传统型	传统型
目录名称	warrior	weapon	team	cube
源程序文件名	warrior.cpp	weapon.cpp	team.cpp	cube.cpp
输入文件名	warrior.in	weapon.in	team.in	cube.in
输出文件名	warrior.out	weapon.out	team.out	cube.out
单测试点时限	1 秒	1 秒	1 秒	1 秒
内存限制	256 MB	256 MB	256 MB	256 MB

注意事项与提醒（请选手务必仔细阅读）：

1. 比赛一共有四道题，题目顺序与难度无关。
2. 参赛选手需要用 `玩家ID` 作为名称创建文件夹，按照 `源程序文件名` 创建每道题的程序文件，并**直接**放在 `玩家ID` 文件夹中。可以参考试题目录下的 `PTYB-000` 文件夹，也可以直接将该文件夹重命名成自己的 `玩家ID` 。
3. 评测时以 `输入文件名` 和 `输出文件名` 作为名称进行文件输入输出。可以参考试题目录下的 `PTYB-000` 文件夹中的样例程序。
4. 所有名称必须使用**英文小写**。
5. 结果的比较方式为全文比较（忽略**行末空格**和**文末换行**）。
6. 程序可使用的栈内存空间限制与题目的内存限制一致。
7. 每道题都提供了 `Linux` 系统格式的样例文件，放在试题目录下名为 `sample` 的文件夹中。
   注意，使用 `Windows` 的 `记事本` 打开样例文件时**可能**会出现格式错误，可以用 `写字板` 或者编程软件打开。
8. 评测环境为 `Win10` ，允许使用万能头文件 `<bits/stdc++.h>` 。
9. 部分测试点拥有**特殊属性**，可以采用针对性的算法求解并获得这些分数，祝取得好成绩！

胖头鱼战士（warrior）

题目描述

你说的对，但是《原鱼》是由狂狼电竞俱乐部自主研发的一款全新开放世界冒险游戏，考虑到游戏已经家喻户晓，所以在此省略两万字的游戏介绍。

正值春节版本，游戏中推出了一名新的翔元素双手键角色“胖头鱼战士”，该角色一共有三个技能：

1. 普通攻击【翔龙在天】：一共分为 n 段，使用第 i 段普通攻击会花费 t_i 秒的时间，造成 d_i 点伤害，并让角色获得 e_i 点怒气值。只有连续使用普通攻击时才会叠加段数，第 n 段普通攻击结束后或者使用了其他技能后再使用普通攻击，都会从第 1 段开始**重新**计算。
2. 怒气爆发【翔空万里】：当怒气值达到 E 时才能使用，使用后怒气值将变为 0 。使用怒气爆发会花费 t_a 秒的时间，造成 d_a 点伤害，并让角色获得增益效果**之后的 m 次普通攻击造成的伤害翻倍**。如果在增益效果结束前再次使用怒气爆发，会**清除**之前的增益效果并**重新获得**增益效果。怒气爆发**使用完毕后**该技能的冷却时间将变成 u_a 秒。
3. 怒气战技【龙翔虎跃】：使用怒气战技会花费 t_b 秒的时间，造成 d_b 点伤害，让角色获得 e_b 点怒气值并减少怒气爆发 w 秒的冷却时间。怒气战技**使用完毕后**该技能的冷却时间将变成 u_b 秒。

友情提示：每个技能的冷却时间都是**独立**计算的，每秒钟所有技能的冷却时间都会减少 1 秒，冷却时间最少为 0 秒，当技能的冷却时间**大于** 0 秒时**不能**使用该技能。任意技能在使用过程中都**不会**被打断，每个技能在**使用完毕后**才能使用下一个技能。

胖头鱼作为狂狼电竞俱乐部的游戏策划，现在需要测试一下新角色的强度，但由于他很懒，所以他只会按照最简单的方式使用技能：可以使用技能时，优先使用怒气爆发，其次使用怒气战技，最后使用普通攻击。

初始时角色的怒气值为 0 ，怒气战技和怒气爆发的冷却时间也均为 0 ，没有增益效果也没有叠加普通攻击的段数。现在胖头鱼想知道，如果想要造成 s 点伤害，至少需要多少秒？为了历练新人，所以胖头鱼将任务分配给了新招的临时工，也就是你。

输入格式

第一行包含两个整数 n 和 s ，表示普通攻击分为 n 段，总共需要造成 s 点伤害。

第二行包含五个整数 E 、 m 、 t_a 、 d_a 和 u_a ，表示怒气值达到 E 后才能使用怒气爆发，获得 m 次普通攻击的增益，花费 t_a 秒时间，造成 d_a 点伤害，冷却时间变成 u_a 秒。

第三行包含五个整数 w 、 e_b 、 t_b 、 d_b 、 u_b ，表示怒气战技减少怒气爆发 w 秒冷却，获得 e_b 点怒气值，花费 t_b 秒时间，造成 d_b 点伤害，冷却时间变成 u_b 秒。

接下来 n 行每行包含三个整数 t_i 、 d_i 和 e_i ，按顺序描述每一段普通攻击，花费 t_i 秒时间，造成 d_i 点伤害，获得 e_i 点怒气值。

输出格式

输出一个整数，表示要造成 s 点伤害需要的最少时间。

数据样例

【样例 1 输入】

存放在 `sample/warrior` 目录下的 `warrior1.in` 文件中。

```
2 22
1 0 1 1 1
3 0 3 4 7
2 3 0
1 2 0
```

【样例 1 输出】

存放在 `sample/warrior` 目录下的 `warrior1.out` 文件中。

```
16
```

【样例 1 解释】

由于所有技能获得的怒气值均为 0，所以一直无法使用怒气爆发。

首先使用怒气战技，花费 3 秒，造成 4 点伤害，该技能冷却时间变成 7 秒。

然后依次使用普通攻击第 $1, 2, 1, 2, 1$ 段，共花费 8 秒，造成 13 点伤害。

此时怒气战技的冷却时间为 0 秒，再次释放怒气战技，花费 3 秒，造成 4 点伤害。

最后使用普通攻击第 1 段，花费 2 秒，造成 3 点伤害。

此时总伤害为 24，达到目标，共花费 16 秒。

【样例 2 输入】

存放在 `sample/warrior` 目录下的 `warrior2.in` 文件中。

```
2 26
6 2 1 1 10
3 3 1 2 5
2 3 4
1 2 5
```

【样例 2 输出】

存放在 `sample/warrior` 目录下的 `warrior2.out` 文件中。

```
13
```

【样例 2 解释】

首先使用怒气战技，花费 1 秒，造成 2 点伤害，获得 3 点怒气值，该技能冷却时间变成 5 秒。

然后使用普通攻击第 1 段，花费 2 秒，造成 3 点伤害，获得 4 点怒气值。

此时怒气值达到 7，使用怒气爆发，怒气值变成 0，花费 1 秒，造成 1 点伤害，获得 2 次普通攻击的增益，该技能冷却时间变成 10 秒。

然后使用增益普通攻击第 1 段，花费 2 秒，造成 6 点伤害，获得 4 点怒气值。

此时怒气战技的冷却时间为 0 秒，再次释放怒气战技，花费 1 秒，造成 2 点伤害，获得 3 点怒气值，并减少了怒气爆发 3 秒的冷却。

然后使用增益普通攻击第 1 段，普通攻击第 $2, 1$ 段，共花费 5 秒，造成 11 点伤害，获得 13 点怒气值。

此时怒气战技和怒气爆发的冷却时间均为 0，怒气值达到 20，优先使用怒气爆发，花费 1 秒，造成 1 点伤害。

此时总伤害为 26，达到目标，共花费 13 秒。

【样例 3 、 4 和 5】

存放在 `sample/warrior` 目录下的 `warrior3.in` 和 `warrior3.out` 、 `warrior4.in` 和 `warrior4.out` 、 `warrior5.in` 和 `warrior5.out` 文件中。

数据说明

对于所有测试点，满足：

- 关于普通攻击段数、增益效果与伤害总量：$1 \le n \le 5$ 、$0 \le m \le 10$ 、$1 \le s \le 10^{18}$；
- 关于技能的耗时、伤害与怒气：$1 \le t_a, t_b, t_i, d_a, d_b, d_i \le 3$ 、$0 \le e_i, e_b, w \le 3$；
- 关于技能的冷却时间与怒气条件：$0 \le u_a \le 30$ 、$0 \le u_b \le 15$ 、$1 \le E \le 80$ 。

子任务及其特殊属性如下：

- 子任务 1 占 10% 的分数，满足 $u_b = e_b = 0$，表示怒气战技的冷却与获得怒气均为 0 。
- 子任务 2 占 20% 的分数，满足 $e_i = e_b = 0$ 、$t_i = t_a = t_b = 1$ 、$s \le 10^6$，表示所有技能均不会获得怒气且耗时为 1，总伤害较少。
- 子任务 3 占 20% 的分数，满足 $t_i = t_a = t_b = 1$ 、$s \le 10^6$，表示所有技能均耗时为 1，总伤害较少。
- 子任务 4 占 30% 的分数，满足 $s \le 10^6$，表示总伤害较少。
- 子任务 5 占 20% 的分数，没有特殊属性。

武器展示（weapon）

题目描述

众所周知，”胖头鱼战士“是知名二字游戏《原鱼》中的一名双手键角色，顾名思义，就是双手各拿一个键盘当做武器的角色，为了展现出新角色的独特之处，同时让玩家有更好的~~氪金~~游戏体验，游戏策划胖头鱼专门为该角色设置了两类武器：左手键和右手键。

每一把武器都有独特的效果，为了便于玩家进行~~消费~~选择，一个合理的武器展示界面是必不可少的。为此，胖头鱼设计了一个相当科学的展示方法，展示界面可以看作是一个宽度为 w 的网格，高度是无限的，因为界面可以通过鼠标滚轮上下滑动。展示界面又分为**左右两边**，分别用于展示左手键和右手键，两个区域的宽度可以由玩家自行调整，但是**总和**必须是 w 。

因为武器是键盘，所以在展示界面中，所有武器的**高度**均为 1 ，但是宽度可能各不相同。两类武器都会按照**给定顺序**从第 1 行开始由左向右**逐个**陈列，由于展示区域的宽度是有限的，所以当某个武器**无法**在当前行**完整展示**时，就会从下一行开始继续按照上述规则进行陈列。由此可知，两边展示区域的宽度均**不能小于**对应种类武器的**最大**宽度。

胖头鱼现在有 n 把左手键，宽度分别为 $a_1, \ldots, a_n$ ，还有 m 把右手键，宽度分别为 $b_1, \ldots, b_m$ 。考虑到在展示时，如果有一边的区域很高，而另一边很矮，会显得非常不美观，所以现在胖头鱼想知道，如何设置两边区域的宽度，才能使得两个区域高度的**较大值**最小。显然，这个简单的任务很适合新招的临时工拿来练手，所以请你尽快完成吧。

输入格式

第一行包含三个整数 n 、 m 和 w ，表示有 n 把左手键， m 把右手键，展示界面的宽度为 w 。

第二行包含 n 个整数 a_i ，按顺序给出每把左手键的宽度。

第三行包含 m 个整数 b_i ，按顺序给出每把右手键的宽度。

输出格式

输出一个整数，表示两个展示区域高度较大值的最小值。

数据样例

【样例 1 输入】

存放在 `sample/weapon` 目录下的 `weapon1.in` 文件中。

```
3 4 10
8 1 1
2 2 2 2
```

【样例 1 输出】

存放在 `sample/weapon` 目录下的 `weapon1.out` 文件中。

```
4
```

【样例 1 解释】

只能将左边区域的宽度设置成 8，此时右边区域的宽度为 2。

左边区域的高度为 2，右边区域的高度为 4，因此答案为 $\max(2,4)=4$。

【样例 2 输入】

存放在 `sample/weapon` 目录下的 `weapon2.in` 文件中。

```
4 4 10
5 2 2 1
2 2 2 2
```

【样例 2 输出】

存放在 `sample/weapon` 目录下的 `weapon2.out` 文件中。

```
2
```

【样例 2 解释】

将左边区域的宽度设置成 6 是最佳方案，此时右边区域的宽度为 4，左右区域的高度均为 2。

【样例 3、4 和 5】

存放在 `sample/weapon` 目录下的 `weapon3.in` 和 `weapon3.out`、`weapon4.in` 和 `weapon4.out`、`weapon5.in` 和 `weapon5.out` 文件中。

数据说明

对于所有测试点，满足 $0\le n,m\le 2\times 10^5$、$1\le a_i,b_i,w\le 10^9$，$\max(a_i)+\max(b_i)\le w$。

子任务及其特殊属性如下：

- 子任务 1 占 10% 的分数，满足 $a_i=b_i=1$，表示所有武器的宽度均为 1。
- 子任务 2 占 15% 的分数，满足 $n,m\le 20$，表示武器数量极少。
- 子任务 3 占 20% 的分数，满足 $n,m\le 200$，表示武器的数量很少。
- 子任务 4 占 10% 的分数，满足 $m=0$，表示没有右手键。
- 子任务 5 占 15% 的分数，满足 $w\le 100$，表示总宽度很小。
- 子任务 6 占 30% 的分数，没有特殊属性。

角色配队（team）

题目描述

为了彰显出《原鱼》与某其他知名二字游戏的不同，游戏策划胖头鱼在配队设计上可谓煞费苦心。

游戏中一共有 n 名角色，每名角色都有力量和智力两种属性，其中第 i 名角色的力量值为 a_i ，智力值为 b_i 。每支队伍不限制人数，但是必须**逐个**将角色加入到队伍中，第一个加入的角色**没有**任何限制，但之后加入的每一名角色，其力量值与智力值都必须**严格小于**前一名加入的角色。

显然，一支队伍的实力应当与各种属性值相关，但由于游戏~~奇葩~~独特的配队规则，导致大部分玩家都在研究如何让队伍的人数更多，这样会显得队伍更加酷炫。~~邪恶~~细心的胖头鱼发现，如果不使用某些角色，会直接导致队伍人数的上限减少，他将这种角色称为“关键角色”，通过提高关键角色的稀有程度，可以让游戏获得更多的~~盈利~~人气。所以，他需要知道哪些角色是关键角色，还愣着干什么，肯定是交给作为临时工的你来做啊。

输入格式

第一行包含一个整数 n ，表示角色的数量。

接下来 n 行，每行包含两个整数 a_i 和 b_i ，依次描述每个角色的力量值与智力值。

输出格式

第一行输出一个整数 c ，表示关键角色的数量。

第二行输出 c 个整数，按照由小到大的顺序输出所有关键角色的编号，之间用空格隔开。

数据样例

【样例 1 输入】

存放在 `sample/team` 目录下的 `team1.in` 文件中。

```
5
3 3
1 2
2 1
4 5
5 4
```

【样例 1 输出】

存放在 `sample/team` 目录下的 `team1.out` 文件中。

```
1
1
```

【样例 1 解释】

队伍中最多可以有 3 名角色，比如第 $2, 1, 4$ 名角色可以依次选入队伍。

一共有 4 种配队方式可以包含 3 名角色，但每一种配队都包含了角色 1，所以不使用角色 1 时队伍人数的上限会减少，角色 1 是关键角色。同时可以发现其他角色都不是关键角色。

【样例 2 输入】

存放在 `sample/team` 目录下的 `team2.in` 文件中。

```
5
3 3
1 1
2 2
4 4
5 5
```

【样例 2 输出】

存放在 `sample/team` 目录下的 `team2.out` 文件中。

```
5
1 2 3 4 5
```

【样例 2 解释】

队伍中最多可以有 5 名角色，可以将所有角色按照 $2, 3, 1, 4, 5$ 的顺序加入队伍，因此所有角色都是关键角色。

【样例 3 和 4】

存放在 `sample/team` 目录下的 `team3.in` 和 `team3.out` 、 `team4.in` 和 `team4.out` 文件中。

数据说明

对于所有测试点，满足 $1 \leq n \leq 2 \times 10^5$ 、 $1 \leq a_i, b_i \leq 10^9$ 。

子任务及其特殊属性如下：

- 子任务 1 占 10% 的分数，满足 $a_i = b_i = 1$，表示所有角色的力量值与智力值均为 1 。
- 子任务 2 占 10% 的分数，满足 $a_i = b_i = i$，表示第 i 名角色的力量值与智力值均为 i 。
- 子任务 3 占 10% 的分数，满足 $n \leq 20$，表示角色的数量极少。
- 子任务 4 占 10% 的分数，满足 $n \leq 300$，表示角色的数量很少。
- 子任务 5 占 10% 的分数，满足 $n \leq 2000$，表示角色的数量较少。
- 子任务 6 占 30% 的分数，满足 $n \leq 5000$，表示角色的数量不多。
- 子任务 7 占 20% 的分数，没有特殊属性。

深境方块（cube）

题目描述

风靡全球的游戏《原鱼》中有一个副本叫做"深境方块"，每半个月刷新一次，完成挑战可以获得大量奖励。

具体来说，副本中有一个**长方体状**的迷宫，从上到下一共有 3 层，可以将其分别称为上层、中层和下层，每一层都有 $n \times m$ 个房间，呈**网格状**分布。**上层**的每个房间与其**四周存在**的房间都是相通的，并且可以通往**中层**的**相邻**房间；**中层**的房间如同过道，只能通往**下层**的**相邻**房间；下层的每个房间与其**四周存在**的房间也都是相通的。

开启挑战后玩家可以**任选**一间**上层**房间作为起点，然后按照上述规则移动，并在通过 k 个**不同**的房间后完成挑战，所有房间都可以多次**重复**进入。

显然，每个房间中都有怪物需要清理，但胖头鱼是游戏策划，角色和装备应有尽有，所以不用担心难度问题。但是为了评估副本的多样性，他想知道有多少种不同的挑战方法，作为临时工的你应该已经具备基本的自觉了吧。

两种挑战方法被认为是不同的，当且仅当它们通过的**房间集合**是不同的，即存在某个房间，恰好有一种挑战方法通过了该房间。

输入格式

只有一行，包含三个整数 n 、 m 和 k ，表示每层迷宫有 $n \times m$ 个房间，需要通过 k 个不同的房间。

输出格式

只有一个整数，表示有多少种不同的挑战方法。

数据样例

【样例 1 输入】

存放在 `sample/cube` 目录下的 `cube1.in` 文件中。

```
2 3 3
```

【样例 1 输出】

存放在 `sample/cube` 目录下的 `cube1.out` 文件中。

```
30
```

【样例 1 解释】

当所有通过房间都在上层时，所有房间一排有 2 种方案，呈拐角状有 $4 \times 2 = 8$ 种方案，共 10 种方案。

当上层通过了 2 个房间且中层通过了 1 个房间时，上层房间有 7 种方案，每种方案都可以从 2 个相邻中层房间中选一个，所以有 $7 \times 2 = 14$ 种方案。

当三层各通过了 1 个房间时，共有 6 种方案。

因此总共有 $10 + 14 + 6 = 30$ 种方案。

【样例 2 输入】

存放在 `sample/cube` 目录下的 `cube2.in` 文件中。

```
1 4 5
```

【样例 2 输出】

存放在 `sample/cube` 目录下的 `cube2.out` 文件中。

```
26
```

【样例 2 解释】

当上层通过了 4 个房间时，只有 1 种方案，且中层可以任选一个房间通过，共 4 种方案。

当上层通过了 3 个房间时，有 2 种对称方案，每种方案可以任选一对中层和下层相邻的房间，共 $2 \times 3 = 6$ 种方案。

当上层通过了 2 个房间时，如果靠边，根据中层房间的选择，共有 $2 \times (1 + 2) = 6$ 种方案；如果居中，则有 $2 \times 2 = 4$ 种方案，共 10 种方案。

当上层通过了 1 个房间时，中层也只能通过对应的相邻房间，根据位置下层房间共有 $1 + 2 + 2 + 1 = 6$ 种方案。

因此总共有 $4 + 6 + 10 + 8 = 26$ 种方案。

【样例 3 和 4】

存放在 `sample/cube` 目录下的 `cube3.in` 和 `cube3.out` 、 `cube4.in` 和 `cube4.out` 文件中。

数据说明

对于所有测试点，满足 $1 \leq n \times m \leq 400$ 、 $1 \leq k \leq 10$ 。

子任务及其特殊属性如下：

- 子任务 1 占 20% 的分数，满足 $n = 1$ ，表示每层房间都是一排。
- 子任务 2 占 30% 的分数，满足 $k \leq 5$ ，表示需要通过的房间数较少。
- 子任务 3 占 30% 的分数，满足 $n \times m \leq 20$ ，表示房间的总数较少。
- 子任务 4 占 20% 的分数，没有特殊属性。