

CCF 全国信息学奥林匹克联赛（NOIP2017）复赛

普及组

（请选手务必仔细阅读本页内容）

一. 题目概况

中文题目名称	成绩	图书管理员	棋盘	跳房子
英文题目与子目录名	score	librarian	chess	jump
可执行文件名	score	librarian	chess	jump
输入文件名	score.in	librarian.in	chess.in	jump.in
输出文件名	score.out	librarian.out	chess.out	jump.out
每个测试点时限	1 秒	1 秒	1 秒	2 秒
测试点数目	10	10	20	10
每个测试点分值	10	10	5	10
附加样例文件	有	有	有	有
结果比较方式	全文比较（过滤行末空格及文末回车）			
题目类型	传统	传统	传统	传统
运行内存上限	256M	256M	256M	256M

二. 提交源程序文件名

对于 C++ 语言	score.cpp	librarian.cpp	chess.cpp	jump.cpp
对于 C 语言	score.c	librarian.c	chess.c	jump.c
对于 pascal 语言	score.pas	librarian.pas	chess.pas	jump.pas

三. 编译命令（不包含任何优化开关）

对于 C++ 语言	g++ -o score score.cpp -lm	g++ -o librarian librarian.cpp -lm	g++ -o chess chess.cpp -lm	g++ -o jump jump.cpp -lm
对于 C 语言	gcc -o score score.c -lm	gcc -o librarian librarian.c -lm	gcc -o chess chess.c -lm	gcc -o jump jump.c -lm
对于 pascal 语言	fpc score.pas	fpc librarian.pas	fpc chess.pas	fpc jump.pas

注意事项：

- 1、文件名（程序名和输入输出文件名）必须使用英文小写。
- 2、C/C++ 中函数 main() 的返回值类型必须是 int，程序正常结束时的返回值必须是 0。
- 3、全国统一评测时采用的机器配置为：CPU AMD Athlon(tm) II x2 240 processor, 2.8GHz, 内存 4G, 上述时限以此配置为准。
- 4、只提供 Linux 格式附加样例文件。
- 5、提交的程序代码文件的放置位置请参照各省的具体要求。
- 6、特别提醒：评测在当前最新公布的 NOI Linux 下进行，各语言的编译器版本以其为准。

1. 成绩

(score.cpp/c/pas)

【问题描述】

牛牛最近学习了 C++ 入门课程，这门课程的总成绩计算方法是：

$$\text{总成绩} = \text{作业成绩} \times 20\% + \text{小测成绩} \times 30\% + \text{期末考试成绩} \times 50\%$$

牛牛想知道，这门课程自己最终能得到多少分。

【输入格式】

输入文件名为 score.in。

输入文件只有 1 行，包含三个非负整数 A、B、C，分别表示牛牛的作业成绩、小测成绩和期末考试成绩。相邻两个数之间用一个空格隔开，三项成绩满分都是 100 分。

【输出格式】

输出文件名为 score.out。

输出文件只有 1 行，包含一个整数，即牛牛这门课程的总成绩，满分也是 100 分。

【输入输出样例 1】

score.in	score.out
100 100 80	90

见选手目录下的 score/score1.in 和 score/score1.ans。

【输入输出样例 1 说明】

牛牛的作业成绩是 100 分，小测成绩是 100 分，期末考试成绩是 80 分，总成绩是 $100 \times 20\% + 100 \times 30\% + 80 \times 50\% = 20 + 30 + 40 = 90$ 。

【输入输出样例 2】

score.in	score.out
60 90 80	79

见选手目录下的 score/score2.in 和 score/score2.ans。

【输入输出样例 2 说明】

牛牛的作业成绩是 60 分，小测成绩是 90 分，期末考试成绩是 80 分，总成绩是 $60 \times 20\% + 90 \times 30\% + 80 \times 50\% = 12 + 27 + 40 = 79$ 。

【数据说明】

对于 30% 的数据， $A = B = 0$ 。

对于另外 30% 的数据， $A = B = 100$ 。

对于 100% 的数据， $0 \leq A, B, C \leq 100$ 且 A、B、C 都是 10 的整数倍。

2. 图书管理员

(librarian.cpp/c/pas)

【问题描述】

图书馆中每本书都有一个图书编码，可以用于快速检索图书，这个图书编码是一个正整数。

每位借书的读者手中有一个需求码，这个需求码也是一个正整数。如果一本书的图书编码恰好以读者的需求码结尾，那么这本书就是这位读者所需要的。

小 D 刚刚当上图书馆的管理员，她知道图书馆里所有书的图书编码，她请你帮她写一个程序，对于每一位读者，求出他所需要的书中图书编码最小的那本书，如果没有他需要的书，请输出-1。

【输入格式】

输入文件名为 librarian.in。

输入文件的第一行，包含两个正整数 n 和 q ，以一个空格分开，分别代表图书馆里书的数量和读者的数量。

接下来的 n 行，每行包含一个正整数，代表图书馆里某本书的图书编码。

接下来的 q 行，每行包含两个正整数，以一个空格分开，第一个正整数代表图书馆里读者的需求码的长度，第二个正整数代表读者的需求码。

【输出格式】

输出文件名为 librarian.out。

输出文件有 q 行，每行包含一个整数，如果存在第 i 个读者所需要的书，则在第 i 行输出第 i 个读者所需要的书中图书编码最小的那本书的图书编码，否则输出-1。

【输入输出样例 1】

librarian.in	librarian.out
5 5	23
2123	1123
1123	-1
23	-1
24	-1
24	
2 23	
3 123	
3 124	
2 12	
2 12	

见选手目录下的 librarian/librarian1.in 和 librarian/librarian1.ans。

【输入输出样例 1 说明】

第一位读者需要的书有 2123、1123、23，其中 23 是最小的图书编码。第二位读者需要的书有 2123、1123，其中 1123 是最小的图书编码。对于第三位，第四位和第五位读者，没有书的图书编码以他们的需求码结尾，即没有他们需要的书，输出-1。

【输入输出样例 2】

见选手目录下的 `librarian/librarian2.in` 和 `librarian/librarian2.ans`。

【数据规模与约定】

对于 20% 的数据， $1 \leq n \leq 2$ 。

另有 20% 的数据， $q = 1$ 。

另有 20% 的数据，所有读者的需求码的长度均为 1。

另有 20% 的数据，所有的图书编码按从小到大的顺序给出。

对于 100% 的数据， $1 \leq n \leq 1,000$ ， $1 \leq q \leq 1,000$ ，所有的图书编码和需求码均不超过 10,000,000。

3. 棋盘

(chess.cpp/c/pas)

【问题描述】

有一个 $m \times m$ 的棋盘，棋盘上每一个格子可能是红色、黄色或没有任何颜色的。你现在要从棋盘的最左上角走到棋盘的最右下角。

任何一个时刻，你所站在的位置必须是有颜色的（不能是无色的），你只能向上、下、左、右四个方向前进。当你从一个格子走向另一个格子时，如果两个格子的颜色相同，那你不需要花费金币；如果不同，则你需要花费 1 个金币。

另外，你可以花费 2 个金币施展魔法让下一个无色格子暂时变为你指定的颜色。但这个魔法不能连续使用，而且这个魔法的持续时间很短，也就是说，如果你使用了这个魔法，走到了这个暂时有颜色的格子上，你就不能继续使用魔法；只有当你离开这个位置，走到一个本来就有颜色的格子上的时候，你才能继续使用这个魔法，而当你离开了这个位置（施展魔法使得变为有颜色的格子）时，这个格子恢复为无色。

现在你要从棋盘的最左上角，走到棋盘的最右下角，求花费的最少金币是多少？

【输入格式】

输入文件名为 chess.in。

数据的第一行包含两个正整数 m, n ，以一个空格分开，分别代表棋盘的大小，棋盘上有颜色的格子的数量。

接下来的 n 行，每行三个正整数 x, y, c ，分别表示坐标为 (x, y) 的格子有颜色 c 。其中 $c=1$ 代表黄色， $c=0$ 代表红色。相邻两个数之间用一个空格隔开。棋盘左上角的坐标为 $(1, 1)$ ，右下角的坐标为 (m, m) 。

棋盘上其余的格子都是无色。保证棋盘的左上角，也就是 $(1, 1)$ 一定是有颜色的。

【输出格式】

输出文件名为 chess.out。

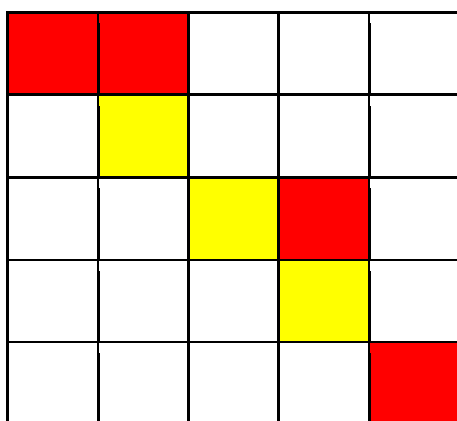
输出一行，一个整数，表示花费的金币的最小值，如果无法到达，输出-1。

【输入输出样例 1】

chess.in	chess.out
5 7 1 1 0 1 2 0 2 2 1 3 3 1 3 4 0 4 4 1 5 5 0	8

见选手目录下的 chess/chess1.in 和 chess/chess1.ans。

【输入输出样例 1 说明】



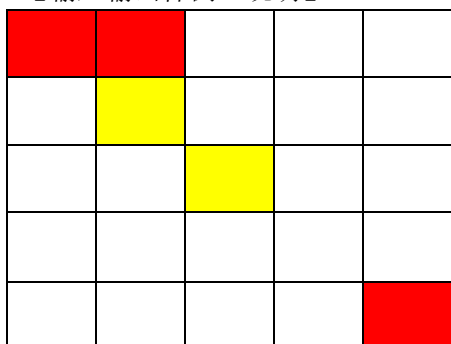
从 (1, 1) 开始, 走到 (1, 2) 不花费金币
 从 (1, 2) 向下走到 (2, 2) 花费 1 枚金币
 从 (2, 2) 施展魔法, 将 (2, 3) 变为黄色, 花费 2 枚金币
 从 (2, 2) 走到 (2, 3) 不花费金币
 从 (2, 3) 走到 (3, 3) 不花费金币
 从 (3, 3) 走到 (3, 4) 花费 1 枚金币
 从 (3, 4) 走到 (4, 4) 花费 1 枚金币
 从 (4, 4) 施展魔法, 将 (4, 5) 变为黄色, 花费 2 枚金币,
 从 (4, 4) 走到 (4, 5) 不花费金币
 从 (4, 5) 走到 (5, 5) 花费 1 枚金币
 共花费 8 枚金币。

【输入输出样例 2】

chess.in	chess.out
5 5 1 1 0 1 2 0 2 2 1 3 3 1 5 5 0	-1

见选手目录下的 chess/chess2.in 和 chess/chess2.ans。

【输入输出样例 2 说明】



从 (1, 1) 走到 (1, 2), 不花费金币

从 (1, 2) 走到 (2, 2), 花费 1 金币

施展魔法将 (2, 3) 变为黄色, 并从 (2, 2) 走到 (2, 3) 花费 2 金币

从 (2, 3) 走到 (3, 3) 不花费金币

从 (3, 3) 只能施展魔法到达 (3, 2), (2, 3), (3, 4), (4, 3)

而从以上四点均无法到达 (5, 5), 故无法到达终点, 输出 -1

【输入输出样例 3】

见选手目录下的 `chess/chess3.in` 和 `chess/chess3.ans`。

【数据规模与约定】

对于 30% 的数据, $1 \leq m \leq 5$, $1 \leq n \leq 10$ 。

对于 60% 的数据, $1 \leq m \leq 20$, $1 \leq n \leq 200$ 。

对于 100% 的数据, $1 \leq m \leq 100$, $1 \leq n \leq 1,000$ 。

4. 跳房子

(jump.cpp/c/pas)

【问题描述】

跳房子，也叫跳飞机，是一种世界性的儿童游戏，也是中国民间传统的体育游戏之一。跳房子的游戏规则如下：

在地面上确定一个起点，然后在起点右侧画 n 个格子，这些格子都在同一条直线上。每个格子内有一个数字（整数），表示到达这个格子能得到的分数。玩家第一次从起点开始向右跳，跳到起点右侧的一个格子内。第二次再从当前位置继续向右跳，依此类推。规则规定：玩家每次都必须跳到当前位置右侧的一个格子内。玩家可以在任意时刻结束游戏，获得的分数为曾经到达过的格子中的数字之和。

现在小 R 研发了一款弹跳机器人来参加这个游戏。但是这个机器人有一个非常严重的缺陷，它每次向右弹跳的距离只能为固定的 d 。小 R 希望改进他的机器人，如果他花 g 个金币改进他的机器人，那么他的机器人灵活性就能增加 g ，但是需要注意的是，每次弹跳的距离至少为 1。具体而言，当 $g < d$ 时，他的机器人每次可以选择向右弹跳的距离为 $d-g, d-g+1, d-g+2, \dots, d+g-2, d+g-1, d+g$ ；否则（当 $g \geq d$ 时），他的机器人每次可以选择向右弹跳的距离为 $1, 2, 3, \dots, d+g-2, d+g-1, d+g$ 。

现在小 R 希望获得至少 k 分，请问他至少要花多少金币来改造他的机器人。

【输入格式】

输入文件名为 jump.in。

第一行三个正整数 n, d, k ，分别表示格子的数目，改进前机器人弹跳的固定距离，以及希望至少获得的分数。相邻两个数之间用一个空格隔开。

接下来 n 行，每行两个正整数 x_i, s_i ，分别表示起点到第 i 个格子的距离以及第 i 个格子的分数。两个数之间用一个空格隔开。保证 x_i 按递增顺序输入。

【输出格式】

输出文件名为 jump.out。

共一行，一个整数，表示至少要花多少金币来改造他的机器人。若无论如何他都无法获得至少 k 分，输出 -1。

【输入输出样例 1】

jump.in	jump.out
7 4 10	2
2 6	
5 -3	
10 3	
11 -3	
13 1	
17 6	
20 2	

见选手目录下的 jump/jump1.in 和 jump/jump1.ans。

【输入输出样例 1 说明】

花费 2 个金币改进后，小 R 的机器人依次选择的向右弹跳的距离分别为 2, 3, 5, 3, 4, 3, 先后到达的位置分别为 2, 5, 10, 13, 17, 20, 对应 1, 2, 3, 5, 6, 7 这 6 个格子。这些格子中的数字之和 15 即为小 R 获得的分数。

【输入输出样例 2】

jump.in	jump.out
7 4 20 2 6 5 -3 10 3 11 -3 13 1 17 6 20 2	-1

见选手目录下的 jump/jump2.in 和 jump/jump2.ans。

【输入输出样例 2 说明】

由于样例中 7 个格子组合的最大可能数字之和只有 18，无论如何都无法获得 20 分。

【输入输出样例 3】

见选手目录下的 jump/jump3.in 和 jump/jump3.ans。

【数据规模与约定】

本题共 10 组测试数据，每组数据 10 分。对于全部的数据满足 $1 \leq n \leq 500000, 1 \leq d \leq 2000, 1 \leq x_i, k \leq 10^9, |s_i| < 10^5$ 。

对于第 1, 2 组测试数据， $n \leq 10$ ；

对于第 3, 4, 5 组测试数据， $n \leq 500$

对于第 6, 7, 8 组测试数据， $d = 1$