Problem A. 搭积木

Input file:	standard input
Output file:	standard output
Time limit:	2 seconds
Memory limit:	64 megabytes

童年时的圣诞节，小摩卡喜欢和兰一起在家里搭积木。

小摩卡有一排长度为 n 的积木墙。在图纸上，第 i 个位置上一共需要搭 h_i 块积木。此时，小摩卡已经在第 i 个位置上了搭了 a_i 块积木。

如果小摩卡在某个位置的积木与图纸相比搭多了，请你提醒她；否则请你告诉她还需多少块积木才能完成图纸上的积木个数要求。

Input

输入第一行包含一个正整数 T $(1 \le T \le 5)$，表示测试数据的组数。

对于每组测试数据，第一行包含 1 个正整数 n $(1 \le n \le 10^5)$，表示积木墙的长度。第二行包含 n 个非负整数 $h_1, h_2, \ldots, h_n$ $(0 \le h_i \le 10^5)$，表示图纸上第 i 个位置需要搭多少积木。第三行包含 n 个非负整数 $a_1, a_2, \ldots, a_n$ $(0 \le a_i \le 10^5)$，表示小摩卡已经在第 i 个位置上搭了多少积木。

数据范围和限制说明:

对于 30% 的数据，$1 \le n \le 10$;

对于 25% 的数据，$1 \le n \le 1000$;

对于 45% 的数据，$1 \le n \le 10^5$。

Output

对于每组测试数据，如果小摩卡目前搭的积木没有搭多，则输出一行包含一个非负整数，表示她还需要搭多少块积木，否则输出一行 -1。

Example

standard input	standard output
2 4 1 2 3 1 0 1 1 0 3 1 1 1 0 1 2	5 -1

Note

部分题目的时间限制**对程序使用的输入输出方式有一定要求**，如果您使用了较慢的输入输出方式，您可能会获得**预期之外的超时结果**。

如果您使用的是 C++ 提供的 iostream 头文件中的 std::cin 和 std::cout 进行输入输出，推荐您在主函数内的第一行添加代码 std::ios::sync_with_stdio(false); ，关闭 iostream 与标准 C 库中的输入输出函数的同步，以提升 std::cin 和 std::cout 的性能。

您也可以使用 cstdio 头文件中提供的标准 C 风格的输入输出方式，使用 scanf(const char* format, ...) 和 printf(const char* format, ...) 函数进行输入输出。

注意：如果您关闭了 iostream 与标准 C 库中的输入输出函数的同步，那么您将不能在同一个程序内混用两种风格的输入输出方式，否则您可能会获得**预期之外的错误结果**。

下面是两段使用 scanf(const char* format, ...) 和 printf(const char* format, ...) 函数进行输入输出的示例代码。

第一段 C++ 示例代码是每行输入两个整数 a 和 b，输出一行 $a+b$ 的结果，直到输入结束。

```
#include <cstdio> // use stdio.h in C
int main() {
  int a, b;
  while (scanf("%d%d", &a, &b) != EOF) {
    printf("%d\n", a + b);
  }
  return 0;
}
```

第二段 C++ 示例代码是每行输入一个长度不超过 127 的不包含空白符的字符串，输出一行 Hello 和字符串里的内容，直到输入结束。

```
#include <cstdio> // use stdio.h in C
char buffer[128];
int main() {
  while (scanf("%s", buffer) != EOF) {
    printf("Hello %s\n", buffer);
  }
  return 0;
}
```

Problem B. 梦想协奏曲

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	64 megabytes

梦想协奏曲是一个音乐节奏游戏，随着乐曲的旋律，屏幕上会不断下落音符，玩家必须在音符掉落到判定线附近时点击音符。根据点击的精确程度，会产生 Perfect，Good，Bad 和 Miss 的判定，根据判定结果你将会获得相应的分数，Perfect 判定获得的分数最多，而 Miss 判定不仅不会获得任何分数，而且会扣除部分血量。

在游玩一首乐曲前，玩家可以选择一张分数加成卡来提高获得的分数。在一首乐曲的某一段时间内，会触发 Fever 阶段，此时分数加成卡将会产生作用，血量也不会减少。假设点击音符获得的基本分数为 x（不同的判定结果会导致基本分数 x 不同），现在有三种不同的分数加成卡：

1. 普通卡：得分额外增加 $100\% \cdot x$；

2. 期间限定卡：若生命值 ≥ 900，得分额外增加 $110\% \cdot x$，否则得分额外增加 $90\% \cdot x$；

3. Festival 限定卡：若判定为 Perfect，得分额外增加 $115\% \cdot x$，否则没有额外加分。

摩卡是一位音游大师。她在进入 Fever 阶段前有 $p\%$ 的概率失误，导致生命值小于 900；在 Fever 阶段期间每个音符他都有 $q\%$ 的概率得到 Perfect 评价。

现在，摩卡希望你帮助她选择一张最佳的卡，使得她期望获得的分数尽量多。如果有多种卡期望获得的分数相同，那么她会优先选择更稀有的卡，即按照 Festival 限定卡，期间限定卡，普通卡的顺序进行选择。

Input

输入第一行包含一个正整数 T $(1 \leq T \leq 10^5)$，表示测试数据的组数。

对于每组测试数据，包含一行 2 个非负整数 p 和 q $(0 \leq p, q \leq 100)$。

Output

对于每组数据，输出一行 "Permanent"表示选择普通卡，"Limited"表示选择期间限定卡，"Festival"表示选择 Festival 限定卡。

Example

standard input	standard output
3 100 0 0 50 0 100	Permanent Limited Festival

Problem C. 类型检查

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	64 megabytes

小摩卡正在学习程序设计语言，她发现程序设计语言中的一些概念与学过的数学知识有很多关联。

在命令式程序设计语言中的代码中，一个函数是一段程序代码块，它可以接收一些参数，执行指令，返回计算结果。在中学的数学课上也学习过函数的概念，函数是定义在两个集合上的二元关系，满足第一个集合（定义域）中的所有元素都对应唯一一个第二个集合（值域）中的元素。她发现程序语言中的函数和数学中的函数非常相似，它们都是接收参数，计算结果。

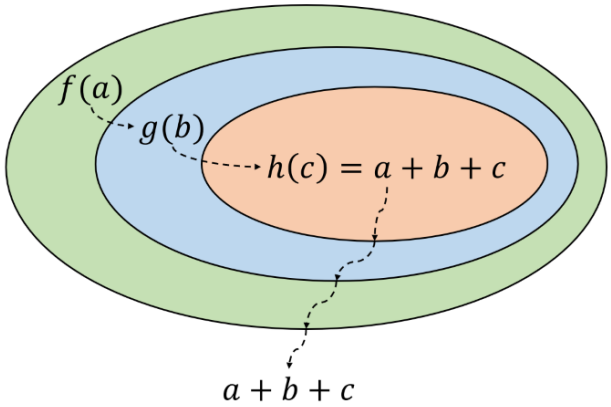
随着小摩卡的深入学习，她发现：如果一个**函数式**程序设计语言**只支持**编写接收**单一参数**的函数，它也能够使用**函数柯里化**的技术，实现接收多个参数进行计算的效果；在数学上，也能够定义一个多元函数，以计算两个自然数之和的函数 $f(x,y)=x+y$ 为例，它实际上只是一个 $\mathbb{N}\times\mathbb{N}$ （两个自然数集合的笛卡尔积）映射到 $\mathbb{N}$ 的函数。

小摩卡希望展示自己的学习成果，她打算向您介绍什么是**函数柯里化**。

定义 **nat** 表示**自然数类型**（nature number 的简写），定义 $\mathbf{nat}\to\mathbf{nat}$ 表示接收一个 **nat** 类型参数然后返回一个 **nat** 类型结果的**函数类型**，也就是在数学上类似一次函数 $f(x)=x+1$。例如，以 `C/C++` 语言为例，`int addOne(int x) { return x + 1; }`。类似地，定义 $\mathbf{nat}\to(\mathbf{nat}\to\mathbf{nat})$ 表示接收一个 **nat** 类型参数然后返回一个 $\mathbf{nat}\to\mathbf{nat}$ 类型结果的**函数类型**，以此类推。定义函数应用，f 是一个类型为 $\mathbf{T}\to\mathbf{U}$ 的函数，x 是一个类型为 $\mathbf{T}$ 的值，那么如果将函数 f 应用到参数 x 上，记作 $f(x)$，$f(x)$ 的类型是 $\mathbf{U}$。

在数学和计算机科学领域，**函数柯里化**是一种将多元函数转化为一元函数序列的方法。例如，有一个计算 3 个参数之和的函数 $add(a,b,c)=a+b+c$。

我们可以将它转化为 3 个函数 f,g,h:



上图中 3 个函数 f,g,h 对应的椭圆形区域，分别表示它们所处的静态词法作用域。在支持**函数是一等公民**的编程语言（例如 `JavaScript`, `Python`, `Haskell` 等）中，可以通过**闭包**的机制访问静态词法作用域内的变量。以此处为例，函数 f 中能够使用的变量有 a，函数 g 中能够使用的变量有 a,b，函数 h 中能够使用的变量有 a,b,c。

$$y = add(a, b, c)$$
$$g = f(a)$$
$$h = g(b)$$
$$y = h(c)$$

其中 f 的类型是 $\mathbf{nat} \to (\mathbf{nat} \to (\mathbf{nat} \to \mathbf{nat}))$，$g$ 的类型是 $\mathbf{nat} \to (\mathbf{nat} \to \mathbf{nat})$，$h$ 的类型是 $\mathbf{nat} \to \mathbf{nat}$，$y$ 的类型是 **nat**。具体地说，函数 f 接收参数 a，返回一个接收 1 个参数的函数 g；函数 g 接收参数 b，返回一个接收 1 个参数的函数 h；函数 h 接收参数 c，返回最终的计算结果 y。

以下给出使用多种不同的程序设计语言，编写柯里化函数 f 计算 $1 + 2 + 3$。

使用 C++ 11 的 Lambda 表达式 (需要包含头文件 <functional>):

```
auto f(int a) {
  return [=](int b) {
    return [=](int c) {
      return a + b + c;
    };
  };
};
// f(1)(2)(3)
```

使用 JavaScript:

```
const f = a => b => c => a + b + c;
// f(1)(2)(3)
```

使用 Python3:

```
f = lambda a : lambda b : lambda c : a + b + c
# f(1)(2)(3)
```

使用 Haskell:

```
f :: Int -> Int -> Int -> Int
f a b c = a + b + c
-- f 1 2 3
```

小摩卡相信您一定学会了函数柯里化的技术，她现在给您出了下面一道题。

小摩卡定义了一个函数 $f(n)$，它将返回一个计算 n 个自然数之和的柯里化函数，$f(n)$ 的返回值的类型是 $\mathbf{nat} \to (\mathbf{nat} \to (\cdots \to \mathbf{nat}))$ （其中一共有 $n + 1$ 个 **nat**）。

小摩卡写下了一个长度为 n 的类型序列，包含 $f(n)$ 和 **nat** 两种类型值。由于 $f(n)$ 之间没有优先级和结合性关系，因此你必须给这个序列加上括号，指定计算的顺序，使得当我们将所有 **nat** 类型替换为具体的自然数值时，我们能够使用添加括号后的表达式正确地得到所有数值的总和；或者你需要报告出小摩卡写下的序列并不合法，即不存在满足条件的括号添加方案。

您能向小摩卡证明自己真的学会函数柯里化了嘛?

Input

输入第一行包含一个正整数 T $(1 \le T \le 10)$，表示测试数据的组数。

对于每组测试数据，第一行包含一个正整数 n $(1 \le n \le 2 \cdot 10^5)$，第二行包含 n 个类型 $type_i$ $(1 \le i \le n)$，其中 $type_i$ 为 **nat** 或 $f(x)$ $(1 \le x \le 10^9)$。

数据范围和限制说明:

对于 15% 的数据，$1 \le n \le 100$;

对于 15% 的数据，$1 \le n \le 5000$;

对于 70% 的数据，$1 \le n \le 2 \cdot 10^5$。

Output

对于每组测试数据，如果存在合法的括号方案输出 "Valid"，否则输出 "Invalid"。

Example

standard input	standard output
6 2 f(1) nat 4 f(2) f(1) nat nat 2 nat f(1) 2 nat nat 2 f(99999) f(1) 2 f(99999) nat	Valid Valid Invalid Invalid Invalid Invalid

Note

对于样例中的第 1 组数据，$f(1)$ 展开后是 **nat** $\to$ **nat**，应用在 **nat** 类型的参数上，得到 **nat**。合法括号方案是 $f(1)$ (**nat**)。

对于样例中的第 2 组数据，$f(2)$ 展开后是 **nat** $\to$ (**nat** $\to$ **nat**)，首先将 $f(1)$ 应用到 **nat** 上得到一个 **nat** 类型，然后 $f(2)$ 应用在这个 **nat** 类型上得到函数 **nat** $\to$ **nat**，即 $f(1)$。最后，将 $f(1)$ 应用到最后一个 **nat** 上，得到 **nat**。合法括号安排方案是 ($f(2)$ ($f(1)$ (**nat**))) (**nat**)。

对于样例中的第 3 组数据，函数应用的正确写法应该是 $f(1)$(**nat**)，因此该类型序列并不合法。

对于样例中的第 4 组数据，没有函数类型，并且剩余了 **2** 个 **nat** 类型的值，因此该类型序列并不合法。

对于样例中的第 5 组数据，没有 **nat** 类型，并且剩余了 2 个函数类型，因此该类型序列并不合法。

对于样例中的第 6 组数据，应用后会得到 $f(99998)$，不是 **nat**，因此该类型序列并不合法。

Problem D. 和风摇滚

Input file:	standard input
Output file:	standard output
Time limit:	2 seconds
Memory limit:	128 megabytes

「Cry cry out，即使笨手笨脚也要挣扎着前进，就连一毫米也不放过，一定要留下脚印」

定义长度 p 是字符串 $s_1s_2\dots s_n$ $(1 \le p \le n$ 且 p 整除 $n)$ 的整周期，当且仅当将 $s_1s_2\dots s_p$ 重复 $\frac{n}{p}$ 次后得到的串与原串相同。形式化地，正整数长度 p 是 s 的整周期，当且仅当 p 整除 $|s|$，且 $\forall p+1 \le i \le |s|$，满足 $s_i = s_{i-p}$。例如，字符串 `abab` 有 2 个整周期，分别为长度为 2 的 `ab` 和长度为 4 的 `abab`；字符串 `abc` 只有一个长度为 3 的整周期 `abc`。

现在，你需要对字符串 $s_1s_2\dots s_n$ 的每个前缀，求出它的整周期个数。

Input

输入第一行包含一个正整数 T $(1 \le T \le 1000)$，表示测试数据的组数。

对于每组测试数据，包含一行仅由**小写英文字母**组成的字符串 $s_1s_2\dots s_n$ $(1 \le n \le 5 \cdot 10^5)$。

保证所有测试数据的字符串长度总和小于等于 $2 \cdot 10^6$。

数据范围和限制说明:

对于 15% 的数据，$1 \le n \le 500$;

对于 15% 的数据，$1 \le n \le 1000$;

对于 40% 的数据，$1 \le n \le 5 \cdot 10^5$;

对于 15% 的数据，$n = 2^{11}$;

对于 15% 的数据，$T = 1000, n = 10$。

Output

对于每组测试数据，输出一行，包含 n 个正整数，分别是长度为 $1, 2, \dots, n$ 的前缀的答案。

注意: **输出没有行末空格**。

Example

standard input	standard output
3 aaaa x abc	1 2 2 3 1 1 1 1

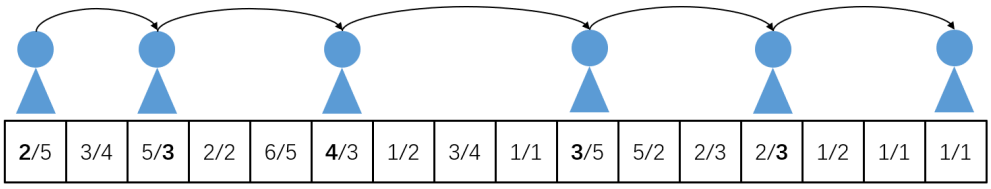
Problem E. 桌游

Input file:	standard input
Output file:	standard output
Time limit:	2 seconds
Memory limit:	256 megabytes

摩卡和兰正在羽泽咖啡店里玩桌游。

这个桌游包含一枚棋子和一块由一排共 n 个方格组成的棋盘。棋盘上方格从左到右依次编号为 1 到 n。每个方格上面标有两个正整数 a_i, b_i $(1 \le a_i, b_i \le n)$。

棋子一开始被放在第 1 个方格内。两名玩家轮流操作棋子，如果棋子位于第 i $(1 \le i \le n)$ 个方格，那么当前玩家可以选择将棋子移动到第 $i + a_i$ 个方格或者第 $i + b_i$ 个方格。注意，棋子移动到的目标方格必须是存在的，即编号小于等于 n。如果第 $i + a_i$ 个方格和 $i + b_i$ 个方格都超出了棋盘范围，那么当前玩家游戏失败；否则游戏继续，轮到下一名玩家继续操作。



以上图为例，一开始棋子位于第 1 个方格内。摩卡首先操作，她可以选择将棋子向右移动 2 格或者 5 格，她觉得游戏一开始移动得越少越好，于是她选择将棋子移动到第 3 个方格内。然后轮到兰进行操作，她决定将棋子向右移动 3 格到达第 6 个方格内。接下来再次轮到摩卡进行操作，她选择向右移动 4 格，到达第 10 个方格。此时，兰发现，不管她选择将棋子移动到第 13 个或者第 15 个方格，摩卡都能一步将棋子移动到最后一个方格，她发现自己输掉了游戏。于是她只能随便选择一个，摩卡获得了游戏的胜利。

她们就这样在羽泽咖啡店玩了一个下午，但是摩卡玩腻了这个游戏。假设摩卡是先手玩家，摩卡和兰都使用最优策略操作。摩卡希望你能直接告诉她是否一定能够取胜，你能帮帮她嘛?

Input

输入第一行包含一个正整数 T $(1 \le T \le 5)$，表示测试数据的组数。

对于每组测试数据，第一行包含一个正整数 n $(1 \le n \le 1000)$，表示棋盘的长度。第二行包含 n 个正整数 $a_1, a_2, \ldots, a_n$ $(1 \le a_i \le n)$。第三行包含 n 个正整数 $b_1, b_2, \ldots, b_n$ $(1 \le b_i \le n)$。

数据范围和限制说明:

对于 20% 的数据，$1 \le n \le 10$;

对于 30% 的数据，$1 \le n \le 100$;

对于 50% 的数据，$1 \le n \le 1000$。

Output

对于每组测试数据，输出一行字符串 `Yes` 或者 `No`，表示摩卡是否先手必胜。

Example

standard input	standard output
2 16 2 3 5 2 6 4 1 3 1 3 5 2 2 1 1 1 5 4 3 2 5 3 2 4 1 5 2 3 3 2 1 1 10 10 1 9 2 9 2 9 4 4 1 6 9 3 9 9 1 1 4 3 6	Yes No

Problem F. 旋律分解

Input file:	standard input
Output file:	standard output
Time limit:	2 seconds
Memory limit:	256 megabytes

给定一个**随机生成**的由小写英文字母组成的字符串。

定义一个字符串 s 的回文串分解是，存在 k 个非空的字符串 $p_1, p_2, \ldots, p_k$，满足 $p_1, p_2, \ldots, p_k$ 都是回文串，且 $p_1, p_2, \ldots, p_k$ 依次连接后得到的串与原串 s 相同。其中，一个字符串 $s_1 s_2 \ldots s_n$ 是回文串，当且仅当 $s_1 s_2 \ldots s_n = s_n s_{n-1} \ldots s_1$，即这个字符串从左往右和从右往左读是一样的，例如 `aba` 是回文串，`abba` 也是回文串，但是 `abbc` 不是回文串（因为 `abbc` $\neq$ `cbba`）。

一个字符串 s 的两个回文串分解方案不同，当且仅当它们划分出来的字符串个数不同，或者存在某些位置的字符串互不相同。

现在，您能帮小摩卡计算一下输入字符串 s 的回文串分解方案数嘛?

Input

输入第一行包含一个正整数 T $(1 \le T \le 10)$，表示测试数据的组数。

对于每组测试数据，第一行包含两个正整数 n, m $(1 \le n \le 2 \cdot 10^6, 2 \le m \le 26)$，表示输入字符串的长度和字符集大小，第二行包含一个由**前 m 个小写英文字母**组成的字符串 s，保证字符串 s 中的每个字符都是**等概率地**从前 m 个小写英文字母中挑选一个得到的。

保证所有测试数据的字符串总长不超过 $5 \cdot 10^6$。

数据范围和限制说明:

对于 10% 的数据，$1 \le n \le 100, 2 \le m \le 26$;

对于 60% 的数据，$1 \le n \le 2 \cdot 10^6, m = 2$;

对于 30% 的数据，$1 \le n \le 2 \cdot 10^6, 3 \le m \le 26$。

Output

对于每组测试数据，输出一行包含一个整数表示答案。由于答案可能非常巨大，你只需要输出答案对 998244353 取模后的结果即可。

Example

standard input	standard output
2 3 2 aba 3 2 aaa	2 4

Note

对于样例中的第 1 组数据，共有 2 种方案:

1. $\underline{a}\ \underline{b}\ \underline{a}$
2. $\underline{aba}$

对于样例中的第 2 组数据，共有 4 种方案:

1. $\underline{a}\ \underline{a}\ \underline{a}$
2. $\underline{aa}\ \underline{a}$
3. $\underline{a}\ \underline{aa}$
4. $\underline{aaa}$

Problem G. 区间操作

Input file:	standard input
Output file:	standard output
Time limit:	2 seconds
Memory limit:	128 megabytes

对于一个给定的正整数 k，定义整数集合上的函数 $f(x)$:

$$f(x) = \begin{cases} x - k & x \geq 2k \\ f(f(x+k+1)) & x < 2k \end{cases}$$

现在，你有一个长度为 n 的数组 $a_1, a_2, \ldots, a_n$ $(1 \leq a_i \leq k)$，有以下 2 种操作:

1. 将 $[l, r]$ 区间内所有数 a_i 变成 $f(a_i)$;
2. 询问 $[l, r]$ 区间内小于等于 k 的数的个数。

Input

输入第一行包含一个正整数 T $(1 \leq T \leq 10)$，表示测试数据的组数。

对于每组测试数据，第一行包含 3 个正整数 n, q, k $(1 \leq n, q, k \leq 10^5)$，分别表示数组的长度，操作总次数和函数的参数。第二行包含 n 个正整数 $a_1, a_2, \ldots, a_n$ $(1 \leq a_i \leq k)$。随后的 q 行，每行包含 3 个正整数 op, l, r $(op = 1$ 或 $2, 1 \leq l \leq r \leq n)$ 分别表示操作的种类和操作区间的范围。

数据范围和限制说明:

对于 20% 的数据，$1 \leq n, q \leq 500$;

对于 20% 的数据，$1 \leq n, q \leq 1000$;

对于 60% 的数据，$1 \leq n, q \leq 10^5$。

Output

对于每组测试数据的每次操作 2，输出一行包含一个正整数表示答案。

Example

standard input	standard output
1 5 4 17 1 2 3 4 5 1 1 2 2 2 4 1 2 4 2 1 5	3 5

Problem H. 火锅

Input file:	standard input
Output file:	standard output
Time limit:	2 seconds
Memory limit:	64 megabytes

Afterglow 乐队的五名成员在训练结束后，准备去商店街上新开的火锅店吃火锅。她们一共点了 n 道菜，因为火锅是一个环形，于是小摩卡想了一个点子，她按照安排好的某种顺序，将点的所有菜都沿着火锅边缘放了进去，使得相邻的两道菜不属于同一种类。

小摩卡注意到，她们点的 n 道菜不一定存在满足她要求的放菜顺序。您能帮助她判断是否存在这样顺序，如果存在帮助她安排这样的一个顺序嘛?

形式化地，给定一个长度为 n 的数组 $a_1, a_2, \ldots, a_n$，将该数组重新排列，构造一个字典序最小的数组满足，对于每个位置 $1 \le i < n$，有 $a_i \neq a_{i+1}$，且 $a_1 \neq a_n$。

两个不同的数组 $a_1, a_2, \ldots, a_n$ 和 $b_1, b_2, \ldots, b_n$，a 数组**字典序小于** b 数组，当且仅当存在一个下标 $1 \le p \le n$，满足 $a_1 = b_1, a_2 = b_2, \ldots, a_{p-1} = b_{p-1}$ 且 $a_p < b_p$。例如，数组 $1, 2, 3$ 字典序小于数组 $1, 2, 4$，数组 $1, 2, 4$ 字典序小于数组 $1, 3, 2$，数组 $1, 3, 2$ 字典序小于数组 $3, 1, 1$。

Input

输入第一行包含一个正整数 T $(1 \le T \le 5000)$，表示测试数据的组数。

对于每组测试数据，第一行包含 1 个正整数 n $(1 \le n \le 2 \times 10^5)$，表示数组的长度。第二行包含 n 个正整数 $a_1, a_2, \ldots, a_n$ $(1 \le a_i \le 200)$。

保证所有测试数据的数组长度总和小于等于 10^6。

数据范围和限制说明:

对于 55% 的数据，$1 \le T \le 5000, 1 \le n \le 10$;

对于 15% 的数据，$1 \le T \le 200, 1 \le n \le 5000$;

对于 30% 的数据，$1 \le n \le 10^5$。

Output

对于每组测试数据，如果存在满足题面要求的数组，则输出一行包含 n 个正整数，表示重新排列后满足要求的字典序最小的答案数组，否则输出一个数字 -1。

Example

standard input	standard output
2 4 1 2 3 3 5 2 3 1 3 3	1 3 2 3 -1

Problem I. 积分卡盖章

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	128 megabytes

小摩卡正在收集山吹面包房的积分卡，每当她在山吹面包房买完面包后，沙绫都会随机地在她积分卡上选择一个位置盖章。印章图案是一个半径为 R 的圆形。但是小摩卡很喜欢积分卡上面的印章，因此她不希望这些印章图案有任何重叠。

于是每次沙绫盖章前，会先告诉她要盖章的位置，如果在这个章对应的圆形区域与之前某个印章有重叠（公共面积大于 0），那么小摩卡会选择不盖这个印章，否则沙绫会帮她在积分卡上的该位置该章。

现在，小摩卡去山吹面包房买了 n 次面包，有了 n 次盖章机会，您能帮助小摩卡判断每次是否能够盖下印章嘛?

Input

输入第一行包含一个正整数 T $(1 \le T \le 100)$，表示测试数据的组数。

对于每组测试数据，第一行包含 2 个正整数 n, R $(1 \le n \le 10^5, 10 \le R \le 100)$，表示盖章的次数和印章的半径。随后 n 行，每行包含 2 个正整数 x_i, y_i $(1 \le x_i, y_i \le 10^5)$，表示第 i 个章的位置，保证盖章坐标 x_i, y_i 都是**等概率地**从 1 到 10^5 内的所有整数中随机生成得到。

保证对于所有测试数据，盖章次数的总和小于等于 10^6。

数据范围和限制说明:

对于 20% 的数据，$1 \le n \le 100$;

对于 20% 的数据，$1 \le n \le 1000$;

对于 60% 的数据，$1 \le n \le 10^5$。

Output

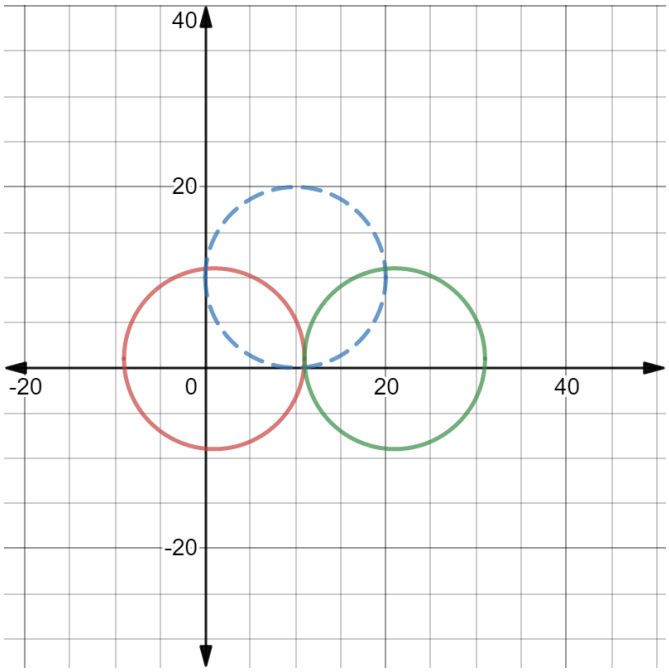
对于每组测试数据，输出 n 行，对于第 i 行，如果能够盖第 i 个章，那么输出 `Yes`，否则输出 `No`。

Example

standard input	standard output
2 3 10 1 1 10 10 21 1 5 100 1 1 200 200 100 100 1000 1 1000 100	Yes No Yes Yes Yes No Yes No

Note

对于样例中的第 1 组数据，如下图所示:



第 1 次盖章位置为红色的圆，成功。

第 2 次盖章位置为蓝色的虚线圆，失败。

第 3 次盖章位置为绿色的圆，成功。

Problem J. LCP 最小生成树

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	64 megabytes

给定一个 n 个点的无向完全图，每个点 i $(1 \le i \le n)$ 上都有一个字符串 s_i。点 i 和点 j 之间边的边权是 s_i 和 s_j 的最长公共前缀的长度。求该无向完全图的最小生成树。

两个字符串 $a_1a_2\ldots a_n$ 和 $b_1b_2\ldots b_m$ 最长公共前缀的长度是最大的 p，满足 $0 \le p \le \min(n,m)$ 且 $a_1a_2\ldots a_p = b_1b_2\ldots b_p$。例如，字符串 `abc` 和 `abd` 的最长公共前缀是 `ab`，长度为 2；字符串 `abc` 和 `bcd` 没有公共前缀，长度为 0。

一个无向联通图 $G = (V, E)$ 的最小生成树是在原图的边集 E 内选出一个大小为 $|V| - 1$ 的子集 E'，满足 E' 使得原图所有点连通并且边集的权值和最小。

Input

输入第一行包含一个正整数 T $(1 \le T \le 10)$，表示测试数据的组数。

对于每组测试数据，第一行包含一个正整数 n $(1 \le n \le 2 \cdot 10^5)$，表示无向完全图的总点数，之后的 n 行每行包含一个由**小写英文字母**组成的字符串 s_i $(1 \le i \le n, 1 \le |s_i| \le 10^5)$。

对于所有数据，保证字符串总长小于等于 $2 \cdot 10^6$。

数据范围和限制说明:

对于 50% 的数据，$1 \le n \le 500$;

对于 20% 的数据，$1 \le n \le 5000$;

对于 30% 的数据，$1 \le n \le 2 \cdot 10^5$。

Output

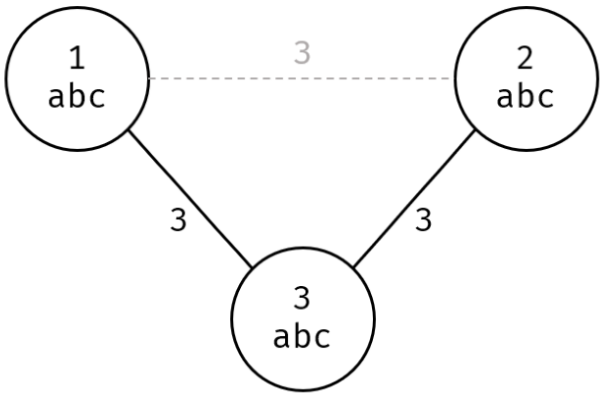
对于每组数据，输出一行，包含一个非负整数，表示这个无向完全图的最小生成树中所有边的权值之和。

Example

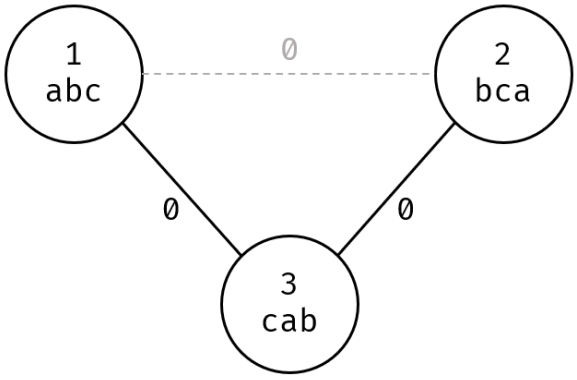
standard input	standard output
3 3 abc abc abc 3 abc bca cab 4 abc abd acd aba	6 0 3

Note

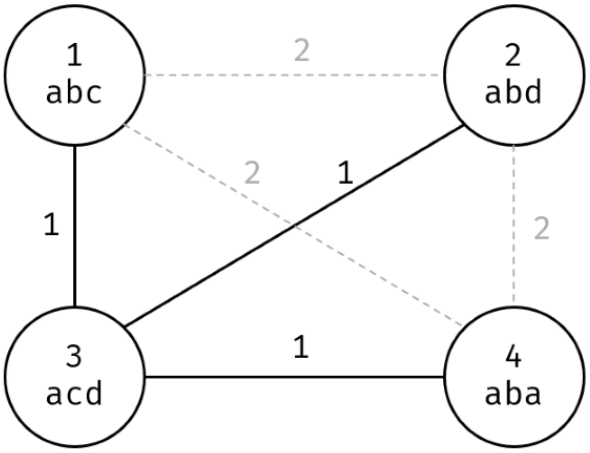
对于样例中的第 1 组数据，一种最小生成树如下图所示：



对于样例中的第 2 组数据，一种最小生成树如下图所示：



对于样例中的第 3 组数据，一种最小生成树如下图所示：



Problem K. 星际巡回演唱会

Input file:	standard input
Output file:	standard output
Time limit:	5 seconds
Memory limit:	64 megabytes

Hello Happy World 乐队即将在整个银河系举办巡回演唱会!

乐队成员米歇尔制定了一份在银河系中 $10^{18}+1$ 个星球巡回演出的路线图，她将路线抽象成了一个以地球为原点，左右各延申 5×10^{17} 个星球的**数轴**。一开始，Hello Happy World 乐队在 0 点（即地球）。因为在星际旅行十分困难，于是黑衣人设计制造了一艘米歇尔飞船，它拥有两种模式：跳跃模式和导航模式。当飞船处于跳跃模式时，它能够按照米歇尔设计的路线（数轴），在星球之间进行跳跃。

如果米歇尔飞船装备有功能为 y $(1\le y_i\le10^5)$ 的跳跃装置并且飞船位于数轴上点 x $(-10^9\le x\le10^9)$ 处的星球上，那么她们能够**立刻**跳跃到点 $x+y$ 或者 $x-y$。只要飞船处于跳跃模式，她们可以在任意时刻任意地点使用飞船上的跳跃装置，没有次数限制。

当飞船切换到导航模式时，**必须选定一个目的星球**，飞船将自动导航到目的星球。在这个过程中，飞船不能装备或者使用跳跃装置，她们也不能举办演唱会。当飞船到达预定的目的星球时，将会**自动切换回跳跃模式**。

在决定举办巡回演出后，将会按顺序发生 n 个事件。只有当前一个事件被完全处理完毕后，下一个事件才会发生。事件分为以下 3 种类型:

1. x_i $happy_i$：乐队打算前往点 x_i 处的星球举办演唱会，如果举办演唱会，她们将会收获 $happy_i$ 的快乐值;
2. y_i：黑衣人为米歇尔飞船制造并装备了一个功能为 y_i 的跳跃装置;
3. pos_i：乐队主唱心心打算切换米歇尔飞船为导航模式，前往点 pos_i 处的星球。

Hello Happy World 乐队希望您能在演出期间作为经理，帮助她们决定是否在某个星球举办演唱会，是否听从乐队主唱心心的想法，以最大化她们收获的快乐值。

Input

输入第一行包含一个正整数 T $(1\le T\le10)$，表示测试数据的组数。

对于每组测试数据，第一行包含一个正整数 n $(1\le n\le2\cdot10^5)$，表示发生的事件数。随后 n 行，每行按照题面说明，首先是一个正整数 $type_i$ $(1\le type_i\le3)$ 表示事件类型。如果 $type_i=1$，跟随着一个整数 x_i 和一个正整数 $happy_i$ $(-10^9\le x_i\le10^9, 1\le happy_i\le10^9)$；如果 $type_i=2$，跟随着一个正整数 y_i $(1\le y_i\le10^5)$；如果 $type_i=3$，跟随着一个整数 pos_i $(-10^9\le pos_i\le10^9)$。

数据范围和限制说明:

对于 10% 的数据，$1\le n\le10$;

对于 30% 的数据，$1\le n\le1000$;

对于 60% 的数据，$1\le n\le2\cdot10^5$。

Output

对于每组数据，输出一行包含一个非负整数，表示她们能够收获的最大快乐值。

Example

standard input	standard output
2 5 1 0 1 2 2 1 1 2 3 1 1 5 3 6 2 4 1 8 1 2 2 3 9 1 1 2 1 2 3	4 4

Note

对于样例中的第 1 组数据:

1. 在点 0 处的星球举办演唱会，获得快乐值 1;
2. 获得功能为 2 的跳跃装置;
3. 无法通过跳跃前往点 1 处的星球举办演唱会;
4. 导航到点 1 处的星球;
5. 连续使用 2 次跳跃装置，到达点 $1+2+2=5$ 处的星球举办演唱会，获得快乐值 3。

最终，Hello Happy World 乐队获得 4 点快乐值。

对于样例中的第 2 组数据:

1. 获得功能为 4 的跳跃装置;
2. 连续使用 2 次跳跃装置，到达点 $0+4+4=8$ 处的星球举办演唱会，获得快乐值 1。
3. 获得功能为 2 的跳跃装置;
4. 放弃导航到点 9 处的星球;
5. 无法通过跳跃前往点 1 处的星球举办演唱会;

6. 连续使用 2 次跳跃装置，到达点 $8-4-2=2$ 处的星球举办演唱会，获得快乐值 3。

最终，Hello Happy World 乐队获得 4 点快乐值。