Зміст

Задача 0.0:1. «games001»

Є одна купка, яка спочатку містить N паличок. Двоє грають у таку гру. Кожен з гравців на кожному своєму ході може забрати з купки або 1, або 2, або 3 палички (але, звісно, не більше, чим їх є в купці). Ніяких інших варіантів ходу нема. Ходять гравці по черзі, пропускати хід не можна. Виграє той, хто забирає останню паличку (можливо, разом із ще однією або ще двома).

Напишіть програму, яка визначатиме, хто виграє при правильній грі обох гравців. Іншими словами, хто може забезпечити собі виграш, хоч би як не грав інший.

Вхідні дані. Єдине ціле число N ($1 \leqslant N \leqslant 12345$) — початкова кількість паличок у купці.

Результати. Єдине ціле число, або `1` (якщо перший гравець може забезпечити собі виграш), або `2` (якщо *другий).*

Приклади:

Клавіатура (stdin)	Екран (stdout)
2	*1*
25	*1*
256	*2*

Задача 0.0:2. «games002»

Це інтерактивна задача. Прочитайте умовну повністю, щоб зрозуміти, як працювати з такими задачами.

Є одна купка, яка спочатку містить N паличок. Двоє грають у таку гру. Кожен з гравців на кожному своєму ході може забрати з купки або 1, або 2, або 3 палички (але, звісно, не більше, чим їх є в купці). Ніяких інших варіантів ходу нема. Ходять гравці по черзі, пропускати хід не можна. Виграє той, хто забирає останню паличку (можливо, разом із ще однією або ще двома).

Напишіть програму, яка інтерактивно гратиме за першого гравця.

Ця задача є інтерактивною: Ваша програма не отримає всіх вхідних даних на початку, а отримуватиме по мірі виконання доуточнення, що залежатимуть від попередніх дій Вашої програми. Тим не менш, *її перевірка буде автоматичною. Тому, слід чітко дотримуватися формату спілкування з програмою, яка грає роль суперника.*

Протокол взаємодії.

На початку, один раз, Ваша програма повинна прочитати одне ціле число в окремому рядку — початкову кількість паличок N ($1 \leqslant N \leqslant 12345$). Потім вона повинна повторювати такий цикл:

1. *Вивести єдине число в окремому рядку — свій хід, тобто кількість паличок, які вона зараз забирає з купки. Це повинно бути ціле число від 1 до 3, причому не більше за поточну кількість паличок у купці.*
2. *Якщо після цього купка стає порожньою, вивести окремим рядком фразу "`I won!`" (без лапок, символ-у-символ згідно зразку) й завершити роботу.*
3. *Інакше, прочитати хід програми-суперниці, тобто кількість паличок, які вона зараз забирає з купки (єдине ціле число, в окремому рядку). Гарантовано, що хід допустимий (є цілим числом від 1 до 3 і не перевищує поточного залишку паличок у купці). Само собою, ця гарантія дійсна лише за умови, що Ваша програма правильно визначила, що гра ще не закінчилася.*

4. *Якщо після цього купка стає порожньою, вивести фразу "`You won...`" (без лапок, символ-у-символ згідно зразку) й завершити роботу.*

Все вищезгадане повинно повторюватися, доки не будуть забрані всі палички (тобто, доки якась із програм-гравців не виграє). Програма-суперниця не виводить фраз "`I won!`" / "`You won...`" чи якихось їх аналогів.

Наполегливо рекомендується, щоб Ваша програма після кожного свого виведення робила дію `flush(output)` *(Pascal), вона ж* `cout.flush()` *(C++), вона ж* `fflush(stdout)` *(C), вона ж* `sys.stdout.flush()` *(Python), вона ж* `System.out.flush()` *(Java). Це істотно зменшує ризик, що проміжна відповідь «загубиться» десь по дорозі, не дійшовши до програми-суперниці.*

Приклад:

Клавіатура (stdin)	Екран (stdout)
5 2	1 2 I won!

Примітка. *Нібито порожні рядки між різними ходами у прикладі зроблені суто для того, щоб краще було видно, хто коли ходить; вводити/виводити їх не треба.*

Загальний хід гри з прикладу можна прокоментувати так. У купці спочатку 5 паличок. Ваша програма забирає одну, лишається 4; програма-суперниця забирає дві, лишається 2; Ваша програма забирає обидві, повідомляє про свій виграш і завершує роботу.

Оцінювання. *У приблизно половині тестів Ваша програма матиме справу з ідеальною програмою-суперницею, яка не робить помилок. У іншій приблизно половині — з різними програмами-суперницями, які грати не вміють — тобто, роблять лише ходи, які дотримуються формальних вимог «забирати лише від 1 до 3 паличок» та «забирати не більше паличок, ніж реально є у купці», але можуть вибирати не найкращий з допустимих ходів, дотримуючись кожна своїх власних уявлень про те, як варто грати в цю гру. Буде оцінюватися як уміння Вашої програми виграти там, де це гарантовано можливо, так і вміння Вашої програми достойно, згідно правил гри, програти, так і вміння Вашої програми скористатися (теж згідно правил) помилками чи іншими неадекватностями програми-суперниці, якщо такі будуть.*

Тобто, щоб отримати абсолютно повні бали, Ваша програма має врахувати все щойно перелічене. Конкретно в цій задачі, можливі часткові бали як у смислі «одні тести успішно пройдені, інші ні», так і в смислі часткових балів за окремо взятий тест. Часткові бали за окремий тест можливі, коли програма грала з дотриманням усіх правил і вчасно завершила гру, але програла, коли можна було виграти, та/або вивела неправильне повідомлення про завершення гри чи не вивела його взагалі, тощо.

За будь-яке порушення правил гри з боку Вашої програми, відповідний тест оцінюватиметься як не пройдений.

Задача 0.0:3. «games003»

порожній, це так треба

Задача 0.0:4. «games004»

На прямокутному полі $N \times M$ клітинок у лівому верхньому кутку стоїть фішка. Ця кутова клітинка гарантовано вільна (не замінована); абсолютно кожна інша клітинка поля може бути хоч вільною, хоч замінованою. Гра полягає в тому, що два гравці поперемінно рухають згадану фішку на якусь кількість клітинок або праворуч, або донизу (кожен гравець сам вирішує, в якому з цих двох напрямків і на скільки клітинок рухати; не можна ні лишати фішку на місці, ні ставати нею в заміновану клітинку, ні перестрибувати нею через замінований клітинку). Програє той, хто не може нікуди походити (і знизу, і праворуч або край поля, або міна). Відповідно, його суперник виграє.

Напишіть програму, яка визначатиме, хто виграє при правильній грі обох гравців. Іншими словами, хто може забезпечити собі виграш, хоч би як не грав інший.

Вхідні дані. Перший рядок містить два цілі числа N та M, розділені одним пропуском (пробілом) — спочатку кількість рядків, потім кількість стовпчиків. Обидва ці значення у межах від 1 до 12.

Далі йдуть N рядків, що задають мінне поле. Кожен з них містить рівно по M символів `.` (позначає вільну клітинку) та/або `*` (позначає замінований клітинку). Ці символи йдуть без роздільників, і кожен з цих N рядків містить лише ці символи та переведення рядка наприкінці.

Результати. Єдине ціле число, або `1` (якщо перший гравець може забезпечити собі виграш), або `2` (якщо другий).

Приклади:

Клавіатура (stdin)	Екран (stdout)

`2 4`
`....`
`.**.`	

`1 1 1` `.`	
`2`	

Примітка. *У першому прикладі, перший гравець може, наприклад, піти на одну клітинку вниз, після чого другому не буде куди йти, і він програє. Якби перший гравець сильно помилився і пішов на першому кроці на три клітинки праворуч, то другий пішов би на одну вниз і виграв. Але це значило б, що перший грає неправильно, а питають про ситуацію правильної гри обох гравців.*

У другому прикладі, першому гравцю відразу ж нема куди йти, і він негайно програє (відповідно, виграє другий гравець). Звісно, з точки зору загальнолюдських уявлень про справедливість це якась дуже нечесна гра…Але вже яка є, описані в умові задачі обмеження формально дотримані.

Задача 0.0:5. «games005»

Єдина відмінність цієї задачі від попередньої — той, кому нема куди ходити, виграє, а не програє. Це єдина з усіх задач серії «Фішка на мінному полі», де вжите таке (протилежне стандартному) правило визначення виграшу.

На прямокутному полі $N \times M$ клітинок у лівому верхньому кутку стоїть фішка. Ця кутова клітинка гарантовано вільна (не замінована); абсолютно кожна інша клітинка поля може бути хоч вільною, хоч замінованою. Гра полягає в тому, що два гравці поперемінно рухають згадану фішку на якусь кількість клітинок праворуч або донизу (кожен гравець сам вирішує, в якому з

цих двох напрямків і на скільки клітинок рухати; не можна ні лишати фішку на місці, ні ставати нею в заміновану клітинку, ні перестрибувати нею через заміновану клітинку). *Виграє* той, хто не може нікуди походити (і знизу, і праворуч або край поля, або міна). Відповідно, його суперник *програє*.

Напишіть програму, яка визначатиме, хто виграє при правильній грі обох гравців. Іншими словами, хто може забезпечити собі виграш, хоч би як не грав інший.

Вхідні дані. Перший рядок містить два цілі числа N та M, розділені одним пропуском (пробілом) — спочатку кількість рядків, потім кількість стовпчиків. Обидва ці значення у межах від 1 до 12.

Далі йдуть N рядків, що задають мінне поле. Кожен з них містить рівно по M символів `.` (позначає вільну клітинку) та/або `*` (позначає заміновану клітинку). Ці символи йдуть без роздільників, і кожен з цих N рядків містить лише ці символи та переведення рядка наприкінці.

Результати. Єдине ціле число, або `1` (якщо перший гравець може забезпечити собі виграш), або `2` (якщо другий).

Приклади:

Клавіатура (stdin)	Екран (stdout)

`2 4`
`....`
`.**.`	

`1 1 1` `.`	
`1`	

Примітка. *У першому прикладі, єдиний спосіб, яким перший гравець може виграти — піти три клітинки праворуч, після чого другий буде змушений піти на одну клітинку вниз (це буде єдиний можливий його хід), після чого першому не буде куди йти, і він виграє. Таким чином, зміна правила визначення переможця на протилежне не гарантує зміни переможця на протилежного: гравці знають змінені правила гри, тож змінюють свої ходи згідно зі зміненими правилами. Водночас, другий приклад показує, що зміна правила визначення переможця на протилежне в деяких випадках може змінити переможця на протилежного.*

Тому, слово «навпаки» вжите в назві задачі для того, щоб підкреслити: дуже часто геть не ясно, що взагалі таке «навпаки». Завершити ж усе це хочемо відомою «цитатою зі шкільного учнівського твору»: «на березі річки доярка доїла корову, а у воді відображалось усе навпаки».

Задача 0.0:6. «games006»

На прямокутному полі $N \times M$ клітинок у лівому верхньому кутку стоїть фішка. Ця кутова клітинка гарантовано вільна (не замінована); абсолютно кожна інша клітинка поля може бути хоч вільною, хоч замінованою. Гра полягає в тому, що два гравці поперемінно рухають згадану фішку на якусь кількість клітинок праворуч або донизу (кожен гравець сам вирішує, в якому з цих двох напрямків і на скільки клітинок рухати; не можна ні лишати фішку на місці, ні ставати нею в заміновану клітинку, ні перестрибувати нею через заміновану клітинку). Програє той, хто не може нікуди походити (і знизу, і праворуч або край поля, або міна). Відповідно, його суперник виграє.

Напишіть програму, яка визначатиме, хто виграє при правильній грі обох гравців. Іншими словами, хто може забезпечити собі виграш, хоч би як не грав інший. Якщо виграє перший, то

програма повинна знайти також сукупність усіх його виграшних перших ходів.

Вхідні дані. Перший рядок містить два цілі числа N та M, розділені одним пропуском (пробілом) — спочатку кількість рядків, потім кількість стовпчиків. Обидва ці значення у межах від 1 до 2023.

Далі йдуть N рядків, що задають мінне поле. Кожен з них містить рівно по M символів `.` (позначає вільну клітинку) та/або `*` (позначає заміновану клітинку). Ці символи йдуть без роздільників, і кожен з цих N рядків містить лише ці символи та переведення рядка наприкінці.

Результати. Перший рядок має містити єдине ціле число, або `1` (якщо перший гравець може забезпечити собі виграш), або `2` (якщо другий). *Якщо відповідь з першого рядка `2`, то на цьому виведення слід припинити. А якщо відповідь з першого рядка `1`, то далі треба вивести також перелік всіх можливих перших ходів першого гравця, після яких другий (при правильній грі першого) вже ніяк не зможе виграти. Цей перелік виводити в такому форматі: спочатку, якщо існує виграшний хід униз, то вивести його як велику латинську літеру `D`, одинарний пропуск (пробіл) та довжину ходу (на скільки клітинок іти вниз); потім, якщо існує виграшний хід направо, то аналогічно, але на початку велику латинську `R`. (Заодно доведіть, що в цій задачі не може бути багатьох виграшних ходів униз чи багатьох виграшних ходів направо.)*

Приклади:

Клавіатура (stdin)	Екран (stdout)

`2 4`
`....`
`.**.`	

`1D 1R 2 1 1` `.`	
`2`	

Примітка. *Спочатку детально проаналізуємо перший приклад.*

Перший гравець може піти `D 1` (на одну клітинку вниз), після чого другому не буде куди йти. А ще перший гравець може піти першим ходом `R 2`; тоді другому не лишиться ніякого вибору, крім як піти `R 1` (на ще одну клітинку праворуч), після чого перший іде `D 1` (вниз), і знову другому не буде куди йти.

Перший хід `R 3` не виграшний, бо після нього другий гравець іде `D 1` і виграє. Перший хід `R 1` теж не виграшний, бо тут може бути як ситуація «другий піде `R 2`, перший `D 1` і виграє», так і ситуація «другий піде `R 1`, перший не матиме іншого вибору, як іти `R 1`, другий піде `D 1` і виграє». Тобто, в разі першого ходу першого гравця `R 1` другий гравець може перехопити ініціативу, якщо зуміє. А питали перелік таких ходів, щоб другий (при правильній грі першого) вже ніяк не міг виграти.

У другому прикладі, першому гравцю відразу ж нема куди йти, і він негайно програє (відповідно, виграє другий гравець).

Задача 0.0:7. «games007»

порожній, це так треба

Задача 0.0:8. «games008»

Це інтерактивна задача. Прочитайте умову повністю, щоб зрозуміти, як працювати з такими задачами.

На прямокутному полі $N \times M$ клітинок у лівому верхньому кутку стоїть фішка. Ця кутова клітинка гарантовано вільна (не замінована); абсолютно кожна інша клітинка поля може бути хоч вільною, хоч замінованою. Гра полягає в тому, що два гравці поперемінно рухають згадану фішку на якусь кількість клітинок праворуч або донизу (кожен гравець сам вирішує, в якому з цих двох напрямків і на скільки клітинок рухати; не можна ні лишати фішку на місці, ні ставати нею в заміновану клітинку, ні перестрибувати нею через заміновану клітинку). Програє той, хто не може нікуди походити (і знизу, і праворуч або край поля, або міна). Відповідно, його суперник виграє.

Напишіть програму, яка інтерактивно гратиме за першого гравця.

Ця задача є інтерактивною: Ваша програма не отримає всіх вхідних даних на початку, а отримуватиме по мірі виконання доуточнення, що залежатимуть від попередніх дій Вашої програми. Тим не менш, *її перевірка буде автоматичною*. Тому, слід чітко дотримуватися формату спілкування з програмою, яка грає роль суперника.

Протокол взаємодії.

На початку, один раз, Ваша програма повинна прочитати поле гри. Перший рядок містить два цілі числа N та M, розділені одним пропуском (пробілом) — спочатку кількість рядків, потім стовпчиків. Обидва ці значення у межах від 1 до 123.

Далі йдуть N рядків, що задають мінне поле. Кожен з них містить рівно по M символів `.` (позначає вільну клітинку) та/або `*` (позначає заміновану клітинку). Ці символи йдуть без роздільників, і кожен з цих N рядків містить лише ці символи та переведення рядка наприкінці.

Потім вона повинна повторювати такий цикл:

1. Якщо йти нема куди, вивести фразу "`I lost...`"(без лапок, символ-у-символ згідно зразку) й завершити роботу.
2. Вивести в окремому рядку свій хід, тобто спочатку або велику латинську `D` (якщо хід униз), або велику латинську `R` (якщо направо), потім пробіл, потім єдине ціле число — на скільки клітинок переміститися. Ця довжина переміщення повинна бути цілою, строго додатною, не виводити за межі поля, не приводити у клітинку з міною і не перестрибувати ні через одну клітинку з міною.
3. Якщо після цього програмі-суперниці нема куди йти, вивести фразу "`I won!`"(без лапок, символ-у-символ згідно зразку) й завершити роботу.
4. Прочитати хід програми-суперниці, у форматі в точності як у позаминулому пункті. Гарантовано, що цей хід відповідає всім вимогам, сформульованим у позаминулому пункті. Само собою, ця гарантія дійсна лише за умови, що Ваша програма правильно визначила, що гра ще не закінчилася.

Все вищезгадане повинно повторюватися, доки фішка не потрапить у клітинку, з якої за правилами не можна вийти (тобто, доки якась із програм-гравців не програє). Програма-суперниця не виводить фраз "`I won!`"/ "`You won...`"чи якихось їх аналогів.

Наполегливо рекомендується, щоб Ваша програма після кожного свого виведення робила дію `flush(output)` (Pascal), вона ж `cout.flush()` (C++), вона ж `fflush(stdout)` (C), вона ж `sys.stdout.flush()` (Python), вона ж `System.out.flush()` (Java). Це істотно зменшує ризик, що проміжна відповідь «загубиться» десь по дорозі, не дійшовши до програми-суперниці.

Приклади:

Клавіатура (stdin)	Екран (stdout)
2 4 .... .**. R 1	 R 2 D 1 I won!
2 4 .... .**.	 D 1 I won!

Примітка.

Нібито порожні рядки між різними ходами у прикладі зроблені суто для того, щоб краще було видно, хто коли ходить; вводити/виводити їх не треба.

Обидві наведені послідовності ходів є прикладами правильної гри. Ваша програма не зобов'язана при різних запусках для одного поля робити різні ходи. Але вона має таке право. При цьому автоматична перевірка не шукатиме кращий чи гірший результат, а просто оцінюватиме перший.

Оцінювання.

У приблизно половині тестів Ваша програма матиме справу з ідеальною програмою-суперницею, яка не робить помилок. У іншій приблизно половині — з різними програмами-суперницями, які грати не вміють — тобто, роблять лише ходи, які дотримуються формальних вимог гри, але можуть вибирати не найкращий з допустимих ходів, дотримуючись кожна своїх власних уявлень про те, як варто грати в цю гру. Буде оцінюватися як уміння Вашої програми виграти там, де це гарантовано можливо, так і вміння Вашої програми достойно, згідно правил гри, програти, так і вміння Вашої програми скористатися (теж згідно правил) помилками чи іншими неадекватностями програми-суперниці, якщо такі будуть.

За будь-яке порушення правил гри з боку Вашої програми, відповідний тест оцінюватиметься як не пройдений.

Приклади:

Задача 0.0:9. «games009»

Є три купки, кожна з яких містить деяку кількість паличок. Двоє грають у таку гру. Кожен з гравців на кожному своєму ході може забрати з будь-якої однієї купки будь-яку кількість паличок, від 1 до зразу всіх паличок цієї купки. Палички можна лише забирати (ні додавати, ні перекладувати з купки в купку не можна). Ніяких інших варіантів ходу нема. Коли купка стає порожньою (кількість паличок=0), гра просто продовжується для решти купок. Ходять гравці по черзі, пропускати хід не можна. Виграє той, хто забирає останню паличку (можливо, разом із ще деякими) з останньої купки. (Інакше кажучи, виграє той, після чийого ходу не лишається жодної палички в жодній купці.)

Напишіть програму, яка визначатиме, хто виграє при правильній грі обох гравців. Іншими словами, хто може забезпечити собі виграш, хоч би як не грав інший.

Вхідні дані. Єдиний рядок містить рівно три цілих числа, кожне в діапазоні від 1 до 40 — початкові кількості паличок у кожній з купок. Серед них можуть бути як однакові, так і різні.

Результати. Єдине ціле число, або `1` (якщо перший гравець може забезпечити собі виграш), або `2` (якщо другий).

Приклади:

Клавіатура (stdin)	Екран (stdout)
2 3 4	1
2 5 5	1
1 2 3	2

Задача 0.0:10. «games010»

Є N купок, кожна з яких містить деяку кількість паличок. Двоє грають у таку гру. Кожен з гравців на кожному своєму ході може забрати з будь-якої однієї купки будь-яку кількість паличок, від 1 до зразу всіх паличок цієї купки. Палички можна лише забирати (ні додавати, ні перекладувати з купки в купку не можна). Ніяких інших варіантів ходу нема. Коли купка стає порожньою (кількість паличок=0), гра просто продовжується для решти купок. Ходять гравці по черзі, пропускати хід не можна. Виграє той, хто забирає останню паличку (можливо, разом із ще деякими) з останньої купки. (Інакше кажучи, виграє той, після чийого ходу не лишається жодної палички в жодній купці.)

Напишіть програму, яка визначатиме, хто виграє при правильній грі обох гравців. Іншими словами, хто може забезпечити собі виграш, хоч би як не грав інший.

Вхідні дані. Перший рядок містить єдине ціле число N ($1 \leqslant N \leqslant 123$) — кількість купок. Другий рядок містить рівно N чисел k_1, k_2, …, k_N, розділених одинарними пропусками (пробілами) — початкові кількості паличок у кожній з купок. Всі чи*с*ла k_1, k_2, …, k_N є цілими, у межах від 1 до 123456, серед них можуть бути як однакові, так і різні.

Результати. Перший рядок має містити єдине ціле число, або `1` (якщо перший гравець може забезпечити собі виграш), або `2` (якщо другий).

Приклади:

Клавіатура (stdin)	Екран (stdout)
2 3 4	1
2 5 5	2
3 1 2 3	2
2 4 8	1

Задача 0.0:11. «games011»

Є N купок, кожна з яких містить деяку кількість паличок. Двоє грають у таку гру. Кожен з гравців на кожному своєму ході може забрати з будь-якої однієї купки будь-яку кількість паличок, від 1 до зразу всіх паличок цієї купки. Палички можна лише забирати (ні додавати, ні перекладувати з купки в купку не можна). Ніяких інших варіантів ходу нема. Коли купка стає порожньою (кількість паличок=0), гра просто продовжується для решти купок. Ходять гравці по черзі, пропускати хід не можна. Виграє той, хто забирає останню паличку (можливо, разом із ще деякими) з останньої купки. (Інакше кажучи, виграє той, після чийого ходу не лишається жодної палички в жодній купці.)

Напишіть програму, яка визначатиме, хто виграє при правильній грі обох гравців. Іншими словами, хто може забезпечити собі виграш, хоч би як не грав інший. Якщо виграє перший, то програма повинна знайти також сукупність усіх його виграшних перших ходів.

Вхідні дані. Перший рядок містить єдине ціле число N ($1 \leqslant N \leqslant 123$) — кількість купок. Другий рядок містить рівно N чисел k_1, k_2, …, k_N, розділених одинарними пропусками (пробілами) — початкові кількості паличок у кожній з купок. Всі числа k_1, k_2, …, k_N є цілими, у межах від 1 до 123456, серед них можуть бути як однакові, так і різні.

Результати. Перший рядок має містити єдине ціле число, або `1` (якщо перший гравець може забезпечити собі виграш), або `2` (якщо другий). Якщо відповідь з першого рядка `2`, то на цьому виведення слід припинити. А якщо відповідь з першого рядка `1`, то далі треба вивести також перелік всіх можливих перших ходів першого гравця, після яких другий (при правильній грі першого) вже ніяк не зможе виграти. Цей перелік виводити в такому форматі: кожен такий хід в окремому рядку; кожен хід записується як пара чисел через пропуск: спочатку з якої купки слід забрати палички, потім скільки штук паличок треба забрати; якщо є багато різних виграшних перших ходів, вони повинні бути відсортовані за зростанням номера купки, а якщо є багато різних виграшних перших ходів, де палички беруться з однієї й тієї ж купки, то за зростанням кількості паличок, що беруться.

Кількість перших виграшних ходів виводити не треба. Вважати, що купки занумеровані з одиниці: 1, 2, …, N.

Приклади:

Клавіатура (stdin)	Екран (stdout)
2 3 4	1 2 1
2 5 5	2
4 1 2 5 7	1 1 1 3 1 4 1

Примітка. В умові є одне дрібне неможливе й непотрібне уточнення. Приблизно у стилі *«якщо* $2+2=5$*, то виведіть слово "хрпщ"»* — його виводити все'дно не доведеться, бо все'дно $2+2 \neq 5$. Знайти це дрібне неможливе й непотрібне уточнення — одне із завдань, які дуже бажано зробити у процесі розв'язування цієї задачі.

Задача 0.0:12. «games012»

Це інтерактивна задача. Прочитайте умовну повністю, щоб зрозуміти, як працювати з такими задачами.

Є N купок, кожна з яких містить деяку кількість паличок. Двоє грають у таку гру. Кожен з гравців на кожному своєму ході може забрати з будь-якої однієї купки будь-яку кількість паличок, від 1 до зразу всіх паличок цієї купки. Палички можна лише забирати (ні додавати, ні перекладувати з купки в купку не можна). Ніяких інших варіантів ходу нема. Коли купка стає порожньою (кількість паличок=0), гра просто продовжується для решти купок. Ходять гравці по черзі, пропускати хід не можна. Виграє той, хто забирає останню паличку (можливо, разом із ще деякими) з останньої купки. (Інакше кажучи, виграє той, після чийого ходу не лишається жодної палички в жодній купці.)

Напишіть програму, яка інтерактивно гратиме за першого гравця.

Ця задача є інтерактивною: Ваша програма не отримає всіх вхідних даних на початку, а отримуватиме по мірі виконання доуточнення, що залежатимуть від попередніх дій Вашої програми. Тим не менш, *її перевірка буде автоматичною*. Тому, слід чітко дотримуватися формату спілкування з програмою, яка грає роль суперника.

Протокол взаємодії.

На початку, один раз, Ваша програма повинна прочитати початкову позицію гри.

Перший рядок містить єдине ціле число N ($1 \leqslant N \leqslant 12$) — кількість купок. Другий рядок містить рівно N чисел k_1, k_2, …, k_N, розділених одинарними пропусками (пробілами) — початкові кількості паличок у кожній з купок. Всі числа k_1, k_2, …, k_N є цілими, у межах від 1 до 1234, серед них можуть бути як однакові, так і різні.

Потім Ваша програма повинна повторювати такий цикл:

1. Вивести два числа, розділені пропуском, у окремому рядку — свій хід, тобто номер купки, з якої вона забирає палички, та кількість паличок, які вона забирає. Це повинно бути ціле число від 1 до поточної кількості паличок у відповідній купці (обидві межі включно). Купки занумеровані з одиниці: 1, 2, …, N. Якщо внаслідок попередніх ходів деякі купки вже стали порожніми, всі купки зберігають свої початкові номери.
2. Якщо це була остання купка й вона після цього стає порожньою, вивести окремим рядком фразу "`I won!`"(без лапок, символ-у-символ згідно зразку) й завершити роботу.
3. Інакше, прочитати хід програми-суперниці, в такому ж форматі: номер купки, з якої вона забирає палички, одинарний пропуск (пробіл), кількість паличок, які вона забирає. Гарантовано, що хід допустимий: купка з таким номером (нумерація з одиниці) існує й містить хоча б одну паличку, а кількість паличок є цілим числом від 1 до поточного залишку паличок у купці (обидві межі включно). Само собою, ця гарантія дійсна лише за умови, що Ваша програма правильно визначила, що гра ще не закінчилася.
4. Якщо це була остання купка й вона після цього стає порожньою, вивести окремим рядком фразу "`You won...`"(без лапок, символ-у-символ згідно зразку) й завершити роботу.

Все вищезгадане повинно повторюватися, доки не будуть забрані всі палички з усіх купок (тобто, доки якась із програм-гравців не виграє). Програма-суперниця не виводить фраз "`I won!`"/ "`You won...`"чи якихось їх аналогів.

Наполегливо рекомендується, щоб Ваша програма після кожного свого виведення робила дію `flush(output)` (Pascal), вона ж `cout.flush()` (C++), вона ж `fflush(stdout)` (C), вона ж `sys.stdout.flush()` (Python), вона ж `System.out.flush()` (Java). Це істотно зменшує ризик, що проміжна відповідь «загубиться» десь по дорозі, не дійшовши до програми-суперниці.

Приклад:

Клавіатура (stdin)	Екран (stdout)
2 3 4 2 2 1 1	2 1 1 2 2 1 I won!

Примітка.

Нібито порожні рядки між різними ходами у прикладі зроблені суто для того, щоб краще було видно, хто коли ходить; вводити/виводити їх не треба.

Загальний хід гри з прикладу можна прокоментувати так. Є дві купки, в одній 3 палички, в іншій 4. Позначимо як (3,4). Ваша програма забирає 1 паличку з купки №2, лишається (3,3); програма-суперниця забирає 2 палички з купки №2, лишається (3,1); Ваша програма забирає 2 палички з купки №1, лишається (1,1); програма-суперниця забирає останню 1 паличку з купки №1, лишається (0,1); Ваша програма забирає останню 1 паличку з останньої купки №2, повідомляє про свій виграш і завершує роботу.

Оцінювання. Оцінювання поблокове, тобто для нарахування балів за блок потрібно, щоб пройшли усі тести цього блоку. Тест з умови при перевірці не використовується. У п'яти блоках тестів (усіх, крім останнього) Ваша програма матиме справу з ідеальною програмою-суперницею, яка не робить помилок.

10% балів припадає на блок № 1, де $N = 1$ (отже, в усіх тестах цього блоку можна і треба виграти).

Ще 15% балів припадає на блок № 2, де $N = 2$.

Ще 15% балів припадає на блок № 3, де $N = 3$.

Ще 15% балів припадає на блок № 4, де $1 \leqslant N \leqslant 12$.

У кожному з блоків №№2–4 є як тести, де Ваша програма може і повинна вигравати, так і тести, де Ваша програма не має шансів виграти і повинна просто достойно згідно правил гри програти.

Ще 10% балів припадає на блок № 5, де $1 \leqslant N \leqslant 12$, причому в жодному з тестів цього блоку виграти не можна, потрібно *лише* достойно згідно правил гри програвати.

Решта 35% балів припадає на останній блок № 6, де Вашій програмі слід мати справу з різними програмами-суперницями, які грати не вміють — роблять ходи, які дотримуються формальних правил гри, але можуть вибирати не найкращий з допустимих ходів, дотримуючись кожна своїх власних уявлень про те, як варто грати в цю гру. Для проходження цього блоку Ваша програма має в кожному з тестів виграти, скориставшися (згідно правил гри) помилками чи іншими неадекватностями програми-суперниці. В цьому блоці № 6 всі тести відносно великі ($7 \leqslant N \leqslant 12$, всі значення k_i від 98 до 1234).

За будь-яке порушення правил гри з боку Вашої програми, відповідний тест (а отже, й увесь відповідний блок) оцінюватиметься як не пройдений.

Задача 0.0:13. «games013»

Ця задача відрізняється від задачі «Нім — 1» в точності тим, що за один хід можна забирати не більше 3-х паличок.

Є N купок, кожна з яких містить деяку кількість паличок. Двоє грають у таку гру. Кожен з гравців на кожному своєму ході може забрати з будь-якої однієї купки або 1, або 2, або 3 палички (але, звісно, не більше, чим їх є в цій купці). Палички можна лише забирати (ні додавати, ні перекладувати з купки в купку не можна). Ніяких інших варіантів ходу нема. Коли купка стає порожньою (кількість паличок=0), гра просто продовжується для решти купок. Ходять гравці по черзі, пропускати хід не можна. Виграє той, хто забирає останню паличку (можливо, разом із ще деякими) з останньої купки. (Інакше кажучи, виграє той, після чийого ходу не лишається жодної палички в жодній купці.)

Напишіть програму, яка визначатиме, хто виграє при правильній грі обох гравців. Іншими словами, хто може забезпечити собі виграш, хоч би як не грав інший.

Вхідні дані. Перший рядок містить єдине ціле число N ($1 \leqslant N \leqslant 123$) — кількість купок. Другий рядок містить рівно N чисел k_1, k_2, …, k_N, розділених одинарними пропусками (пробілами) — початкові кількості паличок у кожній з купок. Всі чи*с*ла k_1, k_2, …, k_N є цілими, у межах від 1 до 123456, серед них можуть бути як однакові, так і різні.

Результати. Перший рядок має містити єдине ціле число, або 1 (якщо перший гравець може забезпечити собі виграш), або 2 (якщо другий).

Приклади:

Клавіатура (stdin)	Екран (stdout)
2 3 4	1
2 5 5	2
3 1 2 3	2
2 4 8	2

Задача 0.0:14. «games014»

Це інтерактивна задача. Прочитайте умовну повністю, щоб зрозуміти, як працювати з такими задачами.

Ця задача відрізняється від задачі «Нім — інтерактивна» в точності тим, що за один хід можна забирати не більше 3-х паличок.

Є N купок, кожна з яких містить деяку кількість паличок. Двоє грають у таку гру. Кожен з гравців на кожному своєму ході може забрати з будь-якої однієї купки або 1, або 2, або 3 палички (але, звісно, не більше, чим їх є в цій купці). Палички можна лише забирати (ні додавати, ні перекладувати з купки в купку не можна). Ніяких інших варіантів ходу нема. Коли купка стає порожньою (кількість паличок=0), гра просто продовжується для решти купок. Ходять гравці по черзі, пропускати хід не можна. Виграє той, хто забирає останню паличку (можливо, разом із ще деякими) з останньої купки. (Інакше кажучи, виграє той, після чийого ходу не лишається жодної палички в жодній купці.)

Напишіть програму, яка інтерактивно гратиме за першого гравця.

Ця задача є інтерактивною: Ваша програма не отримає всіх вхідних даних на початку, а отримуватиме по мірі виконання доуточнення, що залежатимуть від попередніх дій Вашої програми. Тим не менш, *її перевірка буде автоматичною*. Тому, слід чітко дотримуватися формату спілкування з програмою, яка грає роль суперника.

Протокол взаємодії.

На початку, один раз, Ваша програма повинна прочитати початкову позицію гри.

Перший рядок містить єдине ціле число N ($1 \leqslant N \leqslant 12$) — кількість купок. Другий рядок містить рівно N чисел k_1, k_2, …, k_N, розділених одинарними пропусками (пробілами) — початкові кількості паличок у кожній з купок. Всі числа k_1, k_2, …, k_N є цілими, у межах від 1 до 1234, серед них можуть бути як однакові, так і різні.

Потім Ваша програма повинна повторювати такий цикл:

1. Вивести два числа, розділені пропуском, у окремому рядку — свій хід, тобто номер купки, з якої вона забирає палички, та кількість паличок, які вона забирає. Це повинно бути ціле число від 1 до 3 (обидві межі включно) і не перевищувати поточної кількості паличок у відповідній купці. Купки занумеровані з одиниці: 1, 2, …, N. Якщо внаслідок попередніх ходів деякі купки вже стали порожніми, всі купки зберігають свої початкові номери.
2. Якщо це була остання купка й вона після цього стає порожньою, вивести окремим рядком фразу "`I won!`"(без лапок, символ-у-символ згідно зразку) й завершити роботу.
3. Інакше, прочитати хід програми-суперниці, в такому ж форматі: номер купки, з якої вона забирає палички, одинарний пропуск (пробіл), кількість паличок, які вона забирає. Гарантовано, що хід допустимий: купка з таким номером (нумерація з одиниці) існує й містить хоча б одну паличку, а кількість паличок є цілим числом від 1 до поточного залишку паличок у купці (обидві межі включно). Само собою, ця гарантія дійсна лише за умови, що Ваша програма правильно визначила, що гра ще не закінчилася.
4. Якщо це була остання купка й вона після цього стає порожньою, вивести окремим рядком фразу "`You won...`"(без лапок, символ-у-символ згідно зразку) й завершити роботу.

Все вищезгадане повинно повторюватися, доки не будуть забрані всі палички з усіх купок (тобто, доки якась із програм-гравців не виграє). Програма-суперниця не виводить фраз "`I won!`"/ "`You won...`"чи якихось їх аналогів.

Наполегливо рекомендується, щоб Ваша програма після кожного свого виведення робила дію `flush(output)` (Pascal), вона ж `cout.flush()` (C++), вона ж `fflush(stdout)` (C), вона ж `sys.stdout.flush()` (Python), вона ж `System.out.flush()` (Java). Це істотно зменшує ризик, що проміжна відповідь «загубиться» десь по дорозі, не дійшовши до програми-суперниці.

Приклад:

Клавіатура (stdin)	Екран (stdout)
2 4 8 2 3 1 2 2 3	2 1 1 2 2 1 You won...

Примітка.

Нібито порожні рядки між різними ходами у прикладі зроблені суто для того, щоб краще було видно, хто коли ходить; вводити/виводити їх не треба.

Загальний хід гри з прикладу можна прокоментувати так. Є дві купки, в одній 4 палички, в іншій 8. Позначимо як (4,8). Ваша програма забирає 1 паличку з купки №2, лишається (4,7); програма-суперниця забирає 3 палички з купки №2, лишається (4,4); Ваша програма забирає

2 палички з купки №1, лишається (1,1); програма-суперниця забирає останню 1 паличку з купки №1, лишається (0,1); Ваша програма забирає останню 1 паличку з останньої купки №2, повідомляє про свій виграш і завершує роботу.

У звичайному Німі (без обмеження на кількість паличок, які можна забирати за один раз) Вашій програмі варто було б почати з ходу `2 4`, після якого лишилося б (4,4). Однак, у цій задачі не можна забирати відразу 4 палички, тому такий хід неможливий.

Оцінювання. Оцінювання конкретно цієї задачі потестове (бали за кожен тест ставляться окремо й незалежно від інших тестів), програма-суперниця завжди ідеальна (якщо Вашій програмі дісталася програшна позиція, вона може *лише* достойно згідно правил гри програти, а перехопити ініціативу неможливо). Однак, Вам слід забезпечити, щоб Ваша програма в будь-якому разі коректно доводила процес до кінця (виграшу чи програшу).

За будь-яке порушення правил гри з боку Вашої програми, відповідний тест оцінюватиметься як не пройдений.

Задача 0.0:15. «games015»

Нагадаємо деякі істотні для цієї задачі стандартні правила гри в шахи. Грають два гравці, один грає білими, інший чорними. Гра відбувається на шахівниці, тобто дошці 8×8, стовпчики позначаються буквами від “a” до “h” зліва направо, рядки — цифрами від 1 до 8 знизу догори. Кожна клітинка дошки або порожня, або містить одну фігуру. Якщо фігура A (не пішак) може походити згідно з правилами у клітинку, зайняту чужою фігурою B, то внаслідок такого ходу фігуру B б’ють, тобто знімають з дошки. Тому про всі клітинки, куди деяка фігура може походити, кажуть, що вони «під боєм» цієї фігури.

Королю заборонено ходити у клітинки, які перебувають під боєм будь-якої чужої фігури. Якщо один з гравців зробив такий хід, що король суперника опинився під боєм (це називають «шах»), суперник зобов’язаний відповісти таким ходом, щоб його король вже не був під боєм чужої фігури. Якщо такого ходу не існує, то це називають «мат».

Король може ходити на одну клітинку в будь-якому з 8-ми напрямків (ліворуч, праворуч, вперед, назад, в будь-якому напрямку за будь-якою діагоналлю). Ферзь може ходити в будь-якому з цих самих 8-ми напрямків на будь-яку кількість клітинок, але не перетинаючи клітинок, зайнятих фігурами (байдуже, своїми чи чужими).

Нехай на шахівниці розміщено три фігури: білий король, білий ферзь і чорний король. Зараз хід білих. За яку мінімальну кількість ходів вони гарантовано зможуть поставити мат? Чорні робитимуть усе, дозволене правилами гри, щоб уникнути мату або відтермінувати його.

Вхідні дані.

Програма повинна прочитати спочатку кількість тестових блоків T ($1 \leqslant T \leqslant 70000$), потім сам*і* блоки. Кожен блок є окремим рядком, у якому записані позначення трьох клітинок, де розміщені білий король, білий ферзь і чорний король. Позначення клітинки складається із записаних разом букви вертикалі і номера горизонталі, позначення клітинок всередині рядка розділені одиничними пробілами.

Усі задані позиції гарантовано допустимі з точки зору шахових правил (зокрема, чорний король не під боєм).

Результати.

Ваша програма повинна вивести для кожного тесту єдине число — мінімальну кількість ходів. Рахується лише кількість ходів білих (кількість ходів-відповідей чорних не додається).

Приклади:

Клавіатура (stdin)	Екран (stdout)
2 a3 b3 a1 a3 e3 b1	1 2

Задача 0.0:16. «games016»

Одного разу хлопець, що цікавиться олімпіадними задачами з програмування, пішов на прогулянку до лісу. Ходячи поміж деревами лісу, він зустрів Злого Мішку, лісового звіра, що не любить, коли хтось заходить до його лісу. Мішка хотів принести хлопчика в жертву, але дізнавшись, що хлопчик розуміється в програмуванні, вирішив зіграти з ним у гру “Лісові Шахи”.

Гра має наступні правила: на дошці розміром $3 \times N$ у першому рядку дошки знаходяться N чорних пішаків Злого Мішки, у третьому рядку знаходяться N білих пішаків хлопчика, відповідно другий рядок пустий.

Злий Мішка та хлопчик ходять по черзі, починає хлопчик. На кожному кроці гравець обирає пішак, яким або робить хід, або б’є ворожого пішака. Відповідно до правил пішаки ходять на одну клітину вперед, тобто якщо пішак білий, то він може піти з третього рядка на другий, а потім з другого на перший. Якщо пішак чорний, то все навпаки, він може піти з першого рядка на другий, а потім з другого на третій. Звісно, та клітинка, на яку ходить пішак, має бути пустою. Пішак б’є фігури по діагоналі на одну клітинку. В грі “Лісові Шахи” бити фігури є обов’язковим, тобто якщо можна бити фігуру, то її треба обов’язково побити на цьому кроці! Програє той, хто не зможе зробити хід. Ваша задача допомогти хлопчику вказати всі можливі перші ходи, що 100% призведуть до його виграшу, якщо він буде дотримуватись оптимальної стратеґії.

Вхідні дані. Програма читає з клавіатури одне число N ($1 \leqslant N \leqslant 1000$).

Результати. Програма виводить на екран спочатку число K, що є кількістю перших ходів хлопчика, що 100% приведуть його до виграшу, якщо він буде дотримуватись оптимальної стратеґії. Якщо таких немає, тоді треба вивести 0. Далі йдуть K чисел у порядку зростання, що показують, яким за номером білим пішаком повинен піти хлопчик.

Білі пішаки нумеруються від 1 до N зліва направо. Всі числа виводяться через пропуск.

Приклади:

Клавіатура (stdin)	Екран (stdout)
3	1 2
2	2 1 2

Задача 0.0:17. «games017»

Є стрічка шириною в одну клітинку і довжиною в N клітинок. Двоє грають у таку гру. Кожен з гравців на кожному своєму ході може закреслити кілька клітинок. Якщо закреслення відбувається скраю смужки, або безпосередньо поруч з уже закресленою клітинкою, то закреслити

можна або 1, або 2 клітинки підряд. Якщо закреслення відбувається не згідно попереднього речення, а десь всередині досі суцільного фрагмента стрічки (так, щоб по обидва боки від закресленого були незакреслені фрагменти), то закреслити можна або 2, або 4 клітинки підряд. Ніяких інших варіантів ходу нема. Ходять гравці по черзі, пропускати хід не можна. Виграє той, хто закреслює останню клітинку (можливо, разом із ще кількома).

Напишіть програму, яка визначатиме, хто виграє при правильній грі обох гравців. Іншими словами, хто може забезпечити собі виграш, хоч би як не грав інший. Якщо виграє перший, то програма повинна знайти також сукупність усіх його виграшних перших ходів.

Пару прикладів суто для пояснення правил гри.

Повний перелік усіх можливих ходів для позиції «суцільна стрічка з 4-х клітинок»:

З кожного з країв можна закреслити 1 або 2 клітинки; посередині можна закреслити 2 клітинки, але не 4, бо «всередині» вимагає, щоб з кожного з країв хоч щось лишилося. Всього є 5 варіантів ходу.

Повний перелік усіх можливих ходів для позиції «10 клітинок, 3-тя та 4-та вже закреслені раніше»:

(Суцільно-чорні — закреслені раніше; заштриховані — закреслені на поточному ході.) У лівому фрагменті можна закреслити будь-яку одну крайню, або двома різними способами (дві крайні зліва чи дві крайні справа) закреслити обидві. Якщо креслити з країв правого фрагмента, виходить чотири варіанти: закреслити одну зліва, або дві зліва, або одну справа, або дві справа. Якщо креслити всередині правого фрагмента, є один варіант закреслити зразу чотири, і три варіанти закреслити дві. Всього є 11 варіантів ходу.

Вхідні дані.

Єдине ціле число N ($1 \leqslant N \leqslant 2000$) — початкова кількість клітинок у стрічці (спочатку — єдиному неперервному фрагменті).

Результати.

Єдине ціле число, або `1` (якщо перший гравець може забезпечити собі виграш), або `2` (якщо другий). Якщо відповідь з першого рядка `2`, то на цьому виведення слід припинити. А якщо відповідь з першого рядка `1`, то далі треба вивести також перелік всіх можливих перших ходів першого гравця, після яких дру́гий (при правильній грі першого) вже ніяк не зможе виграти. Цей перелік виводити в такому форматі: кожен такий хід в окремому рядку; кожен хід записується як пара чисел через пропуск: номер початкової клітинки закреслення та кількість клітинок, що закреслюються; якщо є багато різних виграшних перших ходів, вони повинні бути відсортовані за зростанням номера початкової клітинки, а якщо є багато різних виграшних перших ходів, де закреслення починаються в одній клітинці, то за зростанням кількості клітинок, що закреслюються.

Кількість перших виграшних ходів виводити не треба. Вважати, що клітинки занумеровані з одиниці: 1, 2, …, N.

Приклади:

Клавіатура (stdin)	Екран (stdout)
2	1 1 2
3	2
4	1 1 1 2 2 4 1
17	1 6 4 7 2 9 4 10 2

Примітка.

У першому прикладі ($N=2$), виграшним є лише один хід: закреслити зразу обидві клітинки й виграти за один напівхід.

У другому прикладі ($N=3$), виграшних ходів нема взагалі, бо виграш може забезпечити 2-й гравець. 1-й гравець не має іншого вибору, крім як креслити чи то одну, чи то дві клітинки біля одного з країв, і в будь-якому з цих випадків 2-му гравцеві дістанеться один неперервний фрагмент або з однієї клітинки, або з двох; в будь-якому разі, його можна увесь закреслити й виграти.

У третьому прикладі ($N=4$), виграшними є формально три різні ходи, два з яких (`1 1` та `4 1`) симетричні: закреслити якусь одну з крайніх клітинок, після чого супернику дістанеться один неперервний фрагмент із трьох клітинок, що призведе до програшу 2-го гравця згідно міркувань попереднього абзацу. Інший виграшний перший хід — закреслити дві клітинки рівно посередині, тоді від стрічки лишається два окремих квадратики 1×1, суперник не матиме іншого вибору, крім як закреслити один з них, після чого перший гравець закреслює останній з цих квадратиків і виграє. Зверніть увагу, що у відповіді ходи *мусять* бути в порядку `1 1`, `2 2`, `4 1`.

Задача 0.0:18. «games018»

Це інтерактивна задача. Прочитайте умовну повністю, щоб зрозуміти, як працювати з такими задачами.

Є стрічка шириною в одну клітинку і довжиною в N клітинок. Двоє грають у таку гру. Кожен з гравців на кожному своєму ході може закреслити кілька клітинок. Якщо закреслення відбувається скраю смужки, або безпосередньо поруч з уже закресленою клітинкою, то закреслити можна або 1, або 2 клітинки підряд. Якщо закреслення відбувається не згідно попереднього речення, а десь всередині досі суцільного фрагмента стрічки (так, щоб по обидва боки від закресленого були незакреслені фрагменти), то закреслити можна або 2, або 4 клітинки підряд. Ніяких інших варіантів ходу нема. Ходять гравці по черзі, пропускати хід не можна. Виграє той, хто закреслює останню клітинку (можливо, разом із ще кількома).

Напишіть програму, яка інтерактивно гратиме за першого гравця.

Ця задача є інтерактивною: Ваша програма не отримає всіх вхідних даних на початку, а отримуватиме по мірі виконання доуточнення, що залежатимуть від попередніх дій Вашої

програми. Тим не менш, *її перевірка буде автоматичною*. Тому, слід чітко дотримуватися формату спілкування з програмою, яка грає роль суперника.

Пару прикладів суто для пояснення правил гри.

Повний перелік усіх можливих ходів для позиції «суцільна стрічка з 4-х клітинок»:

З кожного з країв можна закреслити 1 або 2 клітинки; посередині можна закреслити 2 клітинки, але не 4, бо «всередині» вимагає, щоб з кожного з країв хоч щось лишилося. Всього є 5 варіантів ходу.

Повний перелік усіх можливих ходів для позиції «10 клітинок, 3-тя та 4-та вже закреслені раніше»:

(Суцільно-чорні — закреслені раніше; заштриховані — закреслені на поточному ході.) У лівому фрагменті можна закреслити будь-яку одну крайню, або двома різними способами (дві крайні зліва чи дві крайні справа) закреслити обидві. Якщо креслити з країв правого фрагмента, виходить чотири варіанти: закреслити одну зліва, або дві зліва, або одну справа, або дві справа. Якщо креслити всередині правого фрагмента, є один варіант закреслити зразу чотири, і три варіанти закреслити дві. Всього є 11 варіантів ходу.

Протокол взаємодії.

На початку, один раз, Ваша програма повинна прочитати початкову позицію гри, яка задається єдиним числом N ($1 \leqslant N \leqslant 2000$) — кількість клітинок у єдиній неперервній смужці.

Потім Ваша програма повинна повторювати такий цикл:

1. Вивести два числа, розділені пропуском, у окремому рядку — свій хід, тобто починаючи з якої клітинки вона закреслює, і скільки клітинок, згідно з раніше описаними правилами. Клітинки занумеровані з одиниці (1, 2, …, N), й ці номери лишаються закріпленими за клітинками протягом усієї гри (хоч би як не викреслювали інші клітинки).
2. Якщо при цьому відбулося викреслення останньої клітинки, вивести окремим рядком фразу "`I won!`"(без лапок, символ-у-символ згідно зразку) й завершити роботу.
3. Інакше, прочитати хід програми-суперниці, в такому ж форматі (починаючи з якої клітинки вона закреслює, і скільки клітинок; ці числа подані в одному рядку, розділені одинарним пробілом). Гарантовано, що хід допустимий: відповідає описаним вище правилам, і жодна з нині закреслюваних клітинок не була закреслена раніше. Само собою, ця гарантія дійсна лише за умови, що Ваша програма правильно визначила, що гра ще не закінчилася.
4. Якщо при цьому відбулося викреслення останньої клітинки, вивести окремим рядком фразу "`You won...`"(без лапок, символ-у-символ згідно зразку) й завершити роботу.

Все вищезгадане повинно повторюватися, доки не будуть викреслені всі клітинки (тобто, доки якась із програм-гравців не в*и*грає). Програма-суперниця не виводить фраз "`I won!`"/ "`You won...`"чи якихось їх аналогів.

Наполегливо рекомендується, щоб Ваша програма після кожного свого виведення робила дію `flush(output)` (Pascal), вона ж `cout.flush()` (C++), вона ж `fflush(stdout)` (C), вона ж `sys.stdout.flush()` (Python), вона ж `System.out.flush()` (Java). Це істотно зменшує ризик, що проміжна відповідь «загубиться» десь по дорозі, не дійшовши до програми-суперниці.

Приклади:

Клавіатура (stdin)	Екран (stdout)
7 4 1 1 1	2 2 6 2 5 1 I won!

Примітка.

Нібито порожні рядки між різними ходами у прикладі зроблені суто для того, щоб краще було видно, хто коли ходить; вводити/виводити їх не треба.

Загальний хід гри з прикладу можна прокоментувати так:

коментар	хід	стан стрічки						
до початку гри жодна з 7 клітинок не закреслена		.	.	.	.	.	.	.
Ваша програма закреслює 2-у та 3-ю клітинки	2 2	.	x	x	.	.	.	.
Програма-суперниця закреслює 4-у клітинку	4 1	.	x	x	o	.	.	.
Ваша програма закреслює 6-у та 7-у клітинки	6 2	.	x	x	o	.	x	x
Програма-суперниця закреслює 1-у клітинки	1 1	o	x	x	o	.	x	x
Ваша програма закреслює 5-у клітинку й виграє	5 1	o	x	x	o	x	x	x

Оцінювання. У більшості тестів Ваша програма матиме справу з ідеальною програмою-суперницею, яка не робить помилок. Однак, невелика кількість тестів передбачатиме також і гру з різними програмами-суперницями, які грати не вміють — роблять ходи, які дотримуються формальних правил гри, але можуть вибирати не найкращий з допустимих ходів, дотримуючись кожна своїх власних уявлень про те, як краще грати в цю гру. Буде оцінюватися як уміння Вашої програми виграти там, де це гарантовано можливо, так і вміння Вашої програми достойно, згідно правил гри, програти, так і вміння Вашої програми скористатися (теж згідно правил) помилками чи іншими неадекватностями програми-суперниці, якщо такі будуть. Уміння перехоплювати ініціативу в неідеальної суперниці оцінюватиметься лише на досить великих тестах.

За будь-яке порушення правил гри з боку Вашої програми, відповідний тест оцінюватиметься як не пройдений.

Задача 0.0:19. «games019»

N карток викладені в ряд зліва направо. На кожній картці написане ціле число. Два гравці по черзі забирають по одній картці, причому забирати можна або крайню ліву, або крайню праву. Закінчується гра, коли забрано всі картки (поки картки є, гравець зобов'язаний робити один із можливих ходів). Мета гри — отримати якомога більшу суму (чисел, записаних на забраних картках).

Яку максимальну суму гарантовано зможе набрати перший гравець?

Вхідні дані.

У першому рядку вказано кількість карток N ($1 \leqslant N \leqslant 2013$). У другому рядку через пробіли задані N цілих чисел (що не перевершують за модулем 10^3; інакше кажучи, належать проміжку $[-1000; +1000]$) — значення, записані на картках.

Результати.

Виведіть єдине ціле число — максимальну суму, яку гарантовано зможе набрати перший гравець.

Приклади:

Клавіатура (stdin)	Екран (stdout)
4 1 2 9 3	10

Примітка.

Якщо на першому ході забрати 1, то суперник у відповідь буде змушений забрати або 3, або 2; у будь-якому з цих випадків, перший гравець зможе забрати собі 9, і, таким чином, гарантовано отримати суму 10 (після чого другий гравець забирає останню картку, і гра закінчується).

Якби перший гравець на першому ході забирав 3, то другий міг би у відповідь забрати 9, і в результаті перший отримав би лише $3 + 2 = 5$. При великій дурості, другий гравець міг би відповісти на хід "3"і ходом "1"; у цьому випадку перший гравець міг би отримати суму $3+9 = 12$. Але перший гравець не може гарантувати, що другий зробить такий дурний хід, тому й відповідь не 12, а 10.

Задача 0.0:20. «games020»

порожній, це так треба

Задача 0.0:21. «games021»

N карток викладені в ряд зліва направо. На кожній картці написане ціле число. Два гравці по черзі забирають по одній картці, причому забирати можна лише з правого боку, або 1 картку, або 2 картки, або 3 картки (але, звісно, не більше карток, чим їх лишилося). Закінчується гра, коли забрано всі картки (поки картки є, гравець зобов'язаний робити один із можливих ходів). Мета гри — отримати якомога більшу суму (чисел, записаних на забраних картках).

Яку максимальну суму гарантовано зможе набрати перший гравець?

Вхідні дані.

У першому рядку вказано кількість карток N ($1 \leqslant N \leqslant 1234567$). У другому рядку через пробіли задані N цілих чисел (що не перевершують за модулем 10^3; інакше кажучи, належать проміжку $[-1000; +1000]$) — значення, записані на картках.

Результати.

Виведіть єдине ціле число — максимальну суму, яку гарантовано зможе набрати перший гравець.

Приклади:

Клавіатура (stdin)	Екран (stdout)
7 11 11 11 11 11 11 11	44
8 3 2 999 4 6 7 -5 1	1000
9 23 -17 2 -13 5 3 11 -7 19	30

Примітка.

У першому прикладі, всі числа додатні й абсолютно однакові, тому гравцям вигідно забирати якнайбільше штук карток: 1-й забирає три штуки, потім 2-й забирає три штуки, потім 1-й забирає останню і цим завершує гру з сумою $(11 + 11 + 11) + (11) = 44$.

У другому прикладі, значення 999 сильно перевищує всю решту, тому 1-му гравцю вигідно дбати в першу чергу про те, щоб це найбільше число дісталося саме йому. І перший хід «взяти дві картки 1 та –5» це забезпечує: як би на нього не відповів 2-й (чи забере лише 7, чи дві картки 7 і 6, чи три картки 7, 6 і 4), все'дно на наступному ході 1-й гравець зможе взяти собі зокрема й картку 999. А інші можливі перші ходи 1-го гравця не гарантують йому взяття картки 999: якщо забрати на першому ході три картки, 2-й гравець зможе забрати 999 собі наступним же ходом; якщо забрати на першому ході одну картку, 2-й гравець у відповідь може взяти собі лише одну картку, тобто (згідно з уже наведеними міркуваннями) забезпечити, що картка 999 дістанеться йому. Звісно, для остаточної відповіді потрібно не лише забрати картку 999, а й отримати точне значення суми; це $(1 + (-5)) + (999 + 2 + 3) = 1000$, що досягається при такому сценарії гри: 1-й забирає 1 та (-5); 2-й забирає 7, 6 та 4 (це найкраще, що йому лишається після того, як його позбавили надії взяти 999); 1-й забирає 999, 2 і 3.

Зверніть увагу: щойно розглянутий приклад містить дуже контрінтуїтивний, на перший абсолютно погляд неприродний перший хід: треба взяти від'ємне значення –5 (хоча його можна й не брати), і не взяти після цього додатне значення 7 (хоч його й можна взяти разом з 1 і –5). Однак, усе це нівелюється тим, що саме 1-й гравець може гарантувати, що потім візьме 999.

Очевидно, що при «середніх» значеннях (і різних, і нема яскраво вираженого найбільшого) вищезгадані підходи не працюють, а працює лише специфічне застосування динамічного програмування. Можливість від'ємних значень на картках додатково зменшує ймовірність правильності простих-але-не-завжди-правильних евристик, не заважаючи при цьому динамічному програмуванню.

У третьому прикладі, 30 досягається так: 1-й гравець забирає одну картку 19; 2-й гравець забирає три картки –7, 11, 3; 1-й гравець забирає одну картку 5; 2-й гравець забирає дві картки –13, 2; 1-й гравець забирає дві картки –17, 23, на чому гра закінчується; $(19)+(5)+(-17+23) = 30$.

Задача 0.0:22. «games022»

Це інтерактивна задача. Прочитайте умовну повністю, щоб зрозуміти, як працювати з такими задачами.

N карток викладені в ряд зліва направо. На кожній картці написане ціле число. Два гравці по черзі забирають по одній картці, причому забирати можна лише з правого боку, або 1 картку, або 2 картки, або 3 картки (але, звісно, не більше карток, чим їх лишилося). Закінчується гра, коли забрано всі картки (поки картки є, гравець зобов'язаний робити один із можливих ходів). Мета гри — отримати якомога більшу суму (чисел, записаних на забраних картках).

Напишіть програму, яка інтерактивно гратиме за першого гравця.

Ця задача є інтерактивною: Ваша програма не отримає всіх вхідних даних на початку, а отримуватиме по мірі виконання доуточнення, що залежатимуть від попередніх дій Вашої програми. Тим не менш, *її перевірка буде автоматичною*. Тому, слід чітко дотримуватися формату спілкування з програмою, яка грає роль суперника.

Протокол взаємодії.

На початку, один раз, Ваша програма повинна прочитати початкову позицію гри, яка задається у двох рядках. Перший рядок містить єдине число N ($1 \leqslant N \leqslant 1000$) — кількість карток. У другому рядку через пробіли задані N цілих чисел (що не перевершують за модулем 10^3; інакше кажучи, належать проміжку $[-1000; +1000]$) — значення, записані на картках.

Потім Ваша програма повинна повторювати такий цикл:

1. Вивести окремий рядок, що містить єдине ціле число від 1 до 3 (але не більше, чим поточна кількість карток) — свій хід, який позначає, скільки карток з правого боку забирає Ваша програма.
2. Якщо забрана картка була останньою, вивести окремим рядком фразу формату "`FINISH` *<myScore>* *<yourScore>*"(без лапок; слово `FINISH` треба так і вивести, символ-у-символ згідно зразку; замість *<myScore>* повинна бути сума чисел на картках, забраних Вашою програмою; замість *<yourScore>* повинна бути сума чисел на картках, забраних програмою-суперницею) й завершити роботу.
3. Інакше, прочитати хід програми-суперниці, в такому ж форматі, як описано в позаминулому пункті.
4. Якщо забрана картка була останньою, вивести окремим рядком фразу в абсолютно такому самому форматі, як описано в позаминулому пункті, й завершити роботу.

Все вищезгадане повинно повторюватися, доки не будуть забрані всі картки (тобто, доки гра не завершиться).

Програма-суперниця не виводить фраз формату "`FINISH` *<myScore>* *<yourScore>*"чи якогось аналогічних.

Наполегливо рекомендується, щоб Ваша програма після кожного свого виведення робила дію `flush(output)` (Pascal), вона ж `cout.flush()` (C++), вона ж `fflush(stdout)` (C), вона ж `sys.stdout.flush()` (Python), вона ж `System.out.flush()` (Java). Це істотно зменшує ризик, що проміжна відповідь «загубиться» десь по дорозі, не дійшовши до програми-суперниці.

Приклад:

Клавіатура (stdin)	Екран (stdout)
8 3 2 999 4 6 7 -5 1 3	2 3 FINISH 1000 17
8 3 2 999 4 6 7 -5 1 1 1	2 3 1 FINISH 1008 9

Примітка.

Перший приклад є грою ідеальних суперників, які обидва завжди вибирають найкращі ходи: спочатку 1-й гравець забирає 2 картки (1 та (-5)); потім 2-й забирає 3 картки (7, 6 та 4); потім 1-й забирає 3 картки (999, 2 та 3). Протягом цієї гри 1-й гравець набрав $(1+(-5))+(999+2+3) = 1000$, а 2-й гравець набрав $(7 + 6 + 4) = 17$.

У другому прикладі послідовність карток така сама, але програма-суперниця грає неправильно. Внаслідок цього, Вашій програмі вдається набрати більше: $(1+(-5))+(6+4+999)+(3) = 1008$. Але Ваша програма заранні не знає того, чи оптимально грає програма-суперниця, й повинна бути готова до будь-яких дозволених правилами гри варіантів розвитку подій. Також зверніть увагу, що в цьому прикладі Ваша програма повинна вивести наприкінці `FINISH 1008 9`, згідно фактичної гри (а не попередню теоретичну оцінку цих сум, яка така само `1000 17`. як і в попередньому прикладі).

Оцінювання.

Оцінювання потестове (кожен тест запускається й оцінюється незалежно від результатів проходження інших тестів; оцінка за розв'язок є сумою оцінок за проходження окремих тестів).

Вашій програмі треба буде грати як з ідеальними програмами-суперницями, так і іншими. Вашій програмі при цьому невідомо, чи конкретно в цьому запуску йдеться про ідеального суперника, чи ні. Незалежно від того, якою є програма-суперниця, Ваша програма зобов'язана набрати суму, більшу або рівну сумі, обчисленій за правилами попередньої (не інтерактивної) задачі (якщо Ваша програма набере строго менше, оцінка за відповідний тест буде 0; якщо Ваша програма набере більше, ніяких підвищень балів за це не буде).

Решта вимог до Вашої програми: довести гру до кінця; вчасно помітити, що гра закінчилася, після чого негайно вивести фінальне повідомлення й завершити роботу; правильно порахувати, хто яку суму набрав (фактично).

Задача 0.0:23. «games023»

Кількакрокова послідовна гра на дереві рішень відбувається так: спочатку 1-й гравець повинен прийняти рішення, вибравши один з можливих варіантів свого ходу; деякі з цих варіантів ходу можуть відразу мати числові значення, скільки від того виграв 1-й гравець і скільки 2-й, решта передбачають, що тепер повинен прийняти рішення (вибрати варіант свого ходу) 2-й гравець, вже знаючи, яке рішення перед тим прийняв (який варіант свого ходу вибрав) 1-й гравець. Серед варіантів ходу 2-го гравця теж деякі можуть мати готові числові значення, скільки від того виграв 1-й гравець і скільки 2-й, а решта передбачати, що тепер повинен прийняти рішення (вибрати варіант свого ходу) знову 1-й гравець. І так далі.

(Цю схему можна узагальнити на більшу кількість гравців, але в цій задачі гравців рівно двоє, й рішення приймають спочатку 1-й, потім 2-й, потім 1-й, потім 2-й, …)

Всі гілки такого дерева мусять колись закінчитися «листками» з готовими числовими значеннями, скільки виграв 1-й гравець і скільки 2-й, але шляхи від кореня (початкового вибору 1-го гравця) до різних «листків» є різними, й кількості проміжних вершин-рішень у цих різних шляхах можуть бути різними (однаковими теж можуть).

Вхідні дані. Єдиний рядок, який містить дерево, закодоване таким способом:

1. кожен окремий вузол-«листок» подається у вигляді $(p_1\ p_2)$, тобто: відкривна кругла дужка, величина виграшу 1-го гравця, пробіл, величина виграшу 2-го гравця, закривна кругла дужка;
2. кожен вузол, що не є «листком» (отже, в ньому приймається рішення й різні варіанти ведуть до різних подальших вузлів) подається у вигляді: відкривна квадратна (якщо ходить 1-й гравець) чи кутова (якщо 2-й) дужка, пробіл, код вузла, куди веде перший варіант рішення, пробіл, код вузла, куди веде другий варіант рішення, пробіл, …, код вузла, куди веде останній варіант рішення, пробіл, закривна квадратна чи кутова дужка (так само, `]` для 1-го і `>` для 2-го). При цьому вкладені коди вузлів можуть відповідати чи то попередньому пункту (якщо вони вже не мають розгалужень), чи то поточному (якщо мають).

Наприклад, `[ (2 3) (4 5) (6 1) ]` подає дерево, де єдиний проміжний вузол має розгалуження на три варіанти, які всі є «листками», виграші яких становлять 2 для 1-го і 3 для 2-го, 4 для 1-го і 5 для 2-го, 6 для 1-го і 1 для 2-го.

Гарантовано, що:

1. дерево містить щонайменше один вузол;

2. рядок, що кодує дерево, має не більше 10^5 символів (усіх, включно з пробілами та дужками);
3. для проміжних вузлів, кількість варіантів вибору перебуває в межах від 2 до 6;
4. всі виграші обох гравців (рахуючи по всім вузлам-«листкам») є різними числами, кожне з яких поміщається у 32-бітовий знаковий

Результати. Програма виводить два числа в одному рядку, розділені пропуском: виграші, які мають отримати 1-й та 2-й гравці відповідно, якщо застосувати описаний алгоритм зворотньої індукції.

Приклад:

Клавіатура (stdin)	Екран (stdout)
`[ < [ (-12 50) (-10 1) ] (25 20) > < (-4 -2) (1 4) (40 -10) [ (10 25) (6 -7) ] > (-3 -2) ]`	`25 20`

Примітка.

Практичну цінність&застосовність цього алгоритму сильно знижує поєднання таких факторів:

1. Для застосування цього алгоритму кожен гравець повинен знати все дерево, включно з величинами виграшів у всіх «листках», причому як свого виграшу, так і суперника. Наприклад, коли працівник робить вибір у ситуації «власник призначив низьку ЗП», то вибір варіанту «почати працювати добре, але постійно вимагати підвищити ЗП» обґрунтовується тим, що при його виборі на наступному ході власнику буде вигідніше все-таки підвищити ЗП, чим звільнити працівника, який почав працювати добре. Але якщо працівник думає, що це так, а насправді власник настільки не любить змінювати своє рішення щодо ЗП, що справжні величини виграшу в цьому «листку» не $(+10, +25)$, а $(-10, +25)$, тож власнику краще звільнити такого «бунтівливого» працівника — той самий вибір працівника призвів би вже до його звільнення власником, що працівнику менш вигідно, чим, наприклад, варіант «лінива праця». Навіть якщо працівник чудово знає сукупність варіантів дерева ходів і правильно розраховує величини власних виграшів у кожному вузлі — все це не рятує від невдалого рішення, спричиненого неправильною інформацією про саму лише чужу величину виграшу. Ще одна схожа ситуація в цьому ж дереві: якщо власник думає, що всі величини виграшів такі, як на рисунках, а насправді виграш працівника у ситуації «ЗП добра, працівник працює добре» становить не (+20), а 0, то власнику насправді вигідно призначати низьку ЗП, а не високу (при високій ЗП такий працівник все'дно вибере «працювати ліниво», що власнику невигідно).
2. Застосування цього алгоритму дає чіткий однозначний результат *лише* якщо ніколи не трапиться ситуація, що варіанти, з яких треба обирати, мають однакові оцінки. Нехай, наприклад, варіанти відповіді власника на ситуацію, коли він уже призначив низьку ЗП, а працівник вже почав працювати добре, але вимагати підвищення ЗП, мають виграші $(-6, -7)$ («звільнити») та $(-6, +25)$ («підвищити ЗП»). Тоді працівник взагалі ніяк не може оцінити, чи його вибір працювати добре, але вимагати підвищення ЗП призведе до підвищення ЗП (отже, виграшу (+25)), чи до звільнення (отже, виграшу (–7)). Через це, працівник ніяк не може визначити, чи вибір «працювати добре, але вимагати підвищення ЗП» для нього кращий за вибір «працювати ліниво», чи гірший. Що у свою чергу призводить до того, що взагалі неможливо визначити оцінку для вузла «власник вже взяв на роботу, призначивши низьку ЗП, тепер хід працівника».

   Тому, при використанні цього алгоритму зазвичай вимагають, щоб всі значення виграшу були різними (точніше, всі значення виграшу кожного окремо взятого гравця були різними, бо цей алгоритм ніколи не порівнює виграші різних гравців).

Ще, цьому алгоритму при виборі максимального виграшу серед можливих варіантів по суті байдуже, чи максимальний «трохи більше» за інші, чи «набагато більше». Тому, при використанні цього алгоритму іноді вимагають, щоб для дерева з n «листками» значення виграшу в листках були числами від 1 до n. (Наприклад, якщо маємо відсортовані значення виграшів працівника –10, –3, –2, 0, 1, 4, 20, 25, 50, то, на думку прихильників цієї вимоги, слід замінити –10 на 1, –3 на 2, –2 на 3, 0 на 4, …, 50 на 9.)

Також слід розуміти, що (на відміну від більшості задач комплекту) знайдені величини виграшів *не є* величинами, які кожен окремо гравець може забезпечити собі, хоч би як не грав інший. Алгоритм прямо розраховує на те, що суперник діятиме, намагаючись збільшити свій виграш, а не «як завгодно» і не «аби створити проблеми своєму супернику (нам)». З одного боку, це чудово, бо створює шанси вибрати якийсь із *взаємо*вигідних варіантів (якщо такі існують). Але з іншого боку це означає, що навіть коли нам відоме все дерево рішень, значення усіх оцінок (включаючи чужі) гарантовано правильні, і ми вміємо все обчислити за цим алгоритмом — все'дно знайдена цим алгоритмом оцінка для нашого гравця не гарантована: якщо суперник помилиться чи зіграє безграмотно, він може в результаті погіршити не лише свій виграш, а й наш.

Задача 0.0:24. «games024»

порожній, це так треба

Задача 0.0:25. «games025»

Є одна купка, яка спочатку містить N паличок. Двоє грають у таку гру. Кожен з гравців на кожному своєму ході може забрати з купки деяку кількість паличок (звісно, не більше, чим їх є в купці), **причому кількості паличок, які можна забирати, для різних суперників різні: 1-й гравець може забирати або 4, або 8, або 16, або 32 палички, тоді як 2-й — або 2, або 7, або 14**. Ніяких інших варіантів ходу нема. Ходять гравці по черзі, пропускати хід не можна. Програє той, хто не може походити. Зверніть увагу: оскільки в цій задачі гравці не можуть забирати 1 паличку, можливі також і ситуації, коли палички ще є, а походити вже не можна. Точніше кажучи, 1-й вже не може ходити не лише коли йому не лишили паличок, але також і коли лишили 1, 2 або 3 палички; 2-й вже не може ходити не лише коли йому не лишили паличок, але також і коли лишили 1 паличку.

Вхідні дані. Єдине ціле число N ($1 \leqslant N \leqslant 12345$) — початкова кількість паличок у купці.

Результати. Перший рядок має містити єдине ціле число, або 1 (якщо перший гравець може забезпечити собі виграш), або 2 (якщо другий). Якщо відповідь з першого рядка 2, то на цьому виведення слід припинити. А якщо відповідь з першого рядка 1, то далі треба вивести також перелік всіх можливих перших ходів першого гравця, після яких другий (при правильній грі першого) вже ні*я*к не зможе виграти. Цей перелік виводити в такому форматі: вибрати лише потрібні ч*и*сла з переліку допустимих ходів 4, 8, 16, 32, і записати в другому рядку всі вибрані

(«виграшні») в порядку зростання через пробіли. Кількість «виграшних ходів» виводити не треба і не можна.

Приклади:

Клавіатура (stdin)	Екран (stdout)
7	2
10	1 4

Примітка. У першому тесті, 1-й гравець фактично може забрати лише 4 палички; після цього, у 2-го теж нема вибору (може лише забрати 2 палички), але 2-й походити ще зміг, а 1-й, якому лишається 1 паличка, більше не має ходів і програє.

У другому тесті, у 1-го гравця на першому ході фактично є вибір «забирати 4 чи забирати 8», причому якщо він забере 8, то 2-й забере 2 палички і тим виграє, бо 1-му гравцю нема паличок і тому нема ходів. (Отже, 1-му гравцю не варто забирати 8 на своєму першому ході.) А от якщо 1-й гравець на першому ході здогадається забрати 4 палички (залишиться 6), то у 2-го нема вибору (може лише забрати 2 палички, лишається 4), після чого 1-й гравець може забрати 4 палички і тим виграти, бо 2-му гравцю нема паличок і тому нема ходів.

Задача 0.0:26. «games026»

Розглянемо біматричну гру двох гравців.

Напишіть програму, яка за вказаними матрицями виграшів визначить, в яких парах стратегій кожного з гравців одна домінує іншу.

Деяка стратегія №i *строго домінує* деяку іншу стратегію №k того самого гравця, коли для кожної конкретної стратегії іншого гравця справедливо, що виграш при використанні стратегії №i строго більший, чим при використанні статегії №k.

Інакше кажучи: для 1-го гравця, стратегія №i *строго домінує* деяку іншу стратегію №k тоді й тільки тоді, коли $\forall j (A_{i,j} > A_{k,j})$ (де A — матриця виграшів 1-го гравця); для 2-го гравця, стратегія №j *строго домінує* деяку іншу стратегію №k тоді й тільки тоді, коли $\forall i (B_{i,j} > B_{i,k})$ (де B — матриця виграшів 2-го гравця).

Ще інакше кажучи: для 1-го гравця, стратегія №i *строго домінує* деяку іншу стратегію №k тоді й тільки тоді, коли в матриці виграшів 1-го гравця всі елементи i-го рядка строго більші за відповідні елементи k-го рядка; для 2-го гравця, стратегія №j *строго домінує* деяку іншу стратегію №k тоді й тільки тоді, коли в матриці виграшів 2-го гравця всі елементи j-го стовпчика строго більші за відповідні елементи k-го стовпчика.

Деяка стратегія №i *слабко домінує* деяку іншу стратегію №k того самого гравця, коли для кожної конкретної стратегії іншого гравця справедливо, що виграш при використанні стратегії №i більший або рівний, чим при використанні статегії №k.

Інакше кажучи: для 1-го гравця, стратегія №i *слабко домінує* деяку іншу стратегію №k тоді й тільки тоді, коли $\forall j (A_{i,j} \geqslant A_{k,j})$ (де A — матриця виграшів 1-го гравця); для 2-го гравця, стратегія №j *слабко домінує* деяку іншу стратегію №k тоді й тільки тоді, коли $\forall i (B_{i,j} \geqslant B_{i,k})$ (де B — матриця виграшів 2-го гравця).

Ще інакше кажучи: для 1-го гравця, стратегія №i *слабко домінує* деяку іншу стратегію №k тоді й тільки тоді, коли в матриці виграшів 1-го гравця всі елементи i-го рядка більші або рівні за відповідні елементи k-го рядка; для 2-го гравця, стратегія №j *слабко домінує* деяку іншу

стратегію №k тоді й тільки тоді, коли в матриці виграшів 2-го гравця всі елементи j-го стовпчика більші або рівні за відповідні елементи k-го стовпчика.

Вхідні дані.

У першому рядку через пропуск (пробіл) задано кількість стратегій першого гравця N ($2 \leqslant N \leqslant 12$) та кількість стратегій другого гравця M ($2 \leqslant M \leqslant 12$). Далі йде один порожній рядок. Наступні N рядків містять матрицю виграшів першого гравця, тобто кожен з цих рядків містить рівно M чисел, розділених одинарними пропусками (пробілами), причому j-е число i-го рядка являє собою виграш першого гравця у випадку, якщо перший гравець застосує стратегію №i, а другий стратегію №j. Далі йде один порожній рядок, після якого в аналогічному форматі задано матрицю виграшів другого гравця: теж N рядків по M чисел, де j-е число i-го рядка являє собою виграш другого гравця у випадку, якщо перший гравець застосує стратегію №i, а другий стратегію №j. Всі значення елементів матриць є цілими числами, що не перевищують за модулем (абсолютною величиною) 999.

Результати. Ваша програма повинна вивести спочатку в окремому рядку фразу `1st player`, потім всі можливі пари стратегій i, j (при $i \neq j$) першого гравця, де i-а стратегія домінує j-у стратегію. Якщо домінування строге, то фраза в окремому рядку повинна мати вигляд "i `strictly dominates` j"; якщо строгого домінування нема, але є слабке домінування, то фраза в окремому рядку повинна мати вигляд "i `weakly dominates` j". Виводити слід без лапок, а замість "i" та "j" треба виводити конкретні числа–номери стратегій (вважаючи, що нумерація починається з 1). Всі ці фрази повинні бути відсортовані за зростанням i, а при однакових i — за зростанням j.

Потім слід вивести окремим рядком фразу `2nd player`, а після неї — аналогічні, в такому ж форматі, результати порівнянь стратегій другого гравця.

Приклади:

Клавіатура (stdin)	Екран (stdout)
3 2 2 4 8 6 2 4 11 12 21 22 31 32	1st player 1 weakly dominates 3 2 strictly dominates 1 2 strictly dominates 3 3 weakly dominates 1 2nd player 2 strictly dominates 1

Примітка. Зверніть увагу, що якщо дві стратегії (чи ще більше стратегій) завжди (для всіх можливих стратегій іншого гравця) приносять абсолютно однакові виграші, слід вважати, що вони взаємно слабко домінують одна одну, й виводити це; але аналогічні твердження, що кожна стратегія слабко домінує саму себе, слід пропускати.

Задача 0.0:27. «games027»

Розглянемо узагальнення біматричної гри на трьох гравців: щодо кожного з трьох гравців задано, які стратегії йому доступні, і для кожного гравця є свій тривимірний масив: елемент `a[i][j][k]` масиву `a` виражає виграш першого гравця за умови, що перший гравець вибрав стратегію номер `i`, дру́гий гравець стратегію номер `j`, третій гравець стратегію номер `k`;

елемент `b[i][j][k]` масиву `b` виражає виграш дру́гого гравця за тих самих умов, а елемент `c[i][j][k]` масиву `c` — виграш третього. (Всього є три тривимірні масиви однакових розмірів, аналогічно тому, як у класичних біматричних іграх є дві матриці однакових розмірів.)

Напишіть програму, яка за вказаними тривимірними масивами виграшів визначить, в яких парах стратегій кожного з гравців одна домінує іншу.

Вхідні дані. У першому рядку через пропуски (пробіли) задано кількість стратегій першого гравця N ($2 \leqslant N \leqslant 8$), кількість стратегій другого гравця M ($2 \leqslant M \leqslant 8$) та кількість стратегій третього гравця K ($2 \leqslant K \leqslant 8$). Далі йде один порожній рядок. Наступні $M \cdot K + (M-1)$ рядків містять масив виграшів першого гравця, у вигляді: спочатку рівно M рядків по рівно K чисел, розділених одинарними пропусками (пробілами), причому j-е число i-го рядка являє собою виграш першого гравця у випадку, якщо перший гравець застосує стратегію № 1, другий стратегію № i, а третій стратегію № j. Далі йде один порожній рядок, після якого задані ще рівно M рядків по рівно K чисел, розділених одинарними пропусками (пробілами), де j-е число i-го рядка являє собою виграш першого гравця у випадку, якщо перший гравець застосує стратегію № 2, другий стратегію № i, а третій стратегію № j; і так далі. Після вичерпання всіх $M \cdot K + (M-1)$ рядків (з яких $M \cdot K$ непорожніх та $M-1$ порожніх) слідують три підряд порожні рядки, а після них в аналогічному форматі задано тривимірний масив виграшів другого гравця. Потім слідують ще три підряд порожні рядки, а після них в аналогічному форматі задано тривимірний масив виграшів третього гравця. Всі значення елементів усіх масивів є цілими числами, що не перевищують за модулем (абсолютною величиною) 999.

Результати. Ваша програма повинна вивести спочатку в окремому рядку фразу `1st player`, потім всі можливі пари стратегій i, j (при $i \neq j$) першого гравця, де i-а стратегія домінує j-у стратегію. Якщо домінування строге, то фраза в окремому рядку повинна мати вигляд "i `strictly dominates` j"; якщо строгого домінування нема, але є слабке домінування, то фраза в окремому рядку повинна мати вигляд "i `weakly dominates` j". Виводити слід без лапок, а замість "i" та "j" треба виводити конкретні чи́сла–номери стратегій (вважаючи, що нумерація починається з 1). Всі ці фрази повинні бути відсортовані за зростанням i, а при однакових i — за зростанням j.

Потім слід вивести окремим рядком фразу `2nd player`, а після неї — аналогічні, в такому ж форматі, результати порівнянь стратегій дру́гого гравця.

Потім слід вивести окремим рядком фразу `3rd player`, а після неї — аналогічні, в такому ж форматі, результати порівнянь стратегій третього гравця.

Приклади:

Клавіатура (stdin)	Екран (stdout)
2 2 2 2 2 3 3 1 0 1 2 1 3 1 3 0 2 0 2 0 3 0 1 2 0 0 0	1st player 1 strictly dominates 2 2nd player 1 weakly dominates 2 2 weakly dominates 1 3rd player

Примітка. Якщо 1-ий гравець вибирає свою 1-шу стратегію, то: в разі вибору 1-ої стратегії 2-им гравцем і 1-ої стратегії 3-им гравцем 1-ий отримує 2 (замість 1, якби вибрав 2-гу стратегію); в разі вибору 1-ої стратегії 2-им гравцем і 2-ої стратегії 3-им гравцем 1-ий отримує 2 (замість 0, якби вибрав 2-гу стратегію); в разі вибору 2-ої стратегії 2-им гравцем і 1-ої стратегії 3-им гравцем

1-ий отримує 3 (замість 1, якби вибрав 2-гу стратегію); в разі вибору 2-ої стратегії 2-им гравцем і 2-ої стратегії 3-им гравцем 1-ий отримує 3 (замість 2, якби вибрав 2-гу стратегію). Таким чином, 1-ша стратегія строго домінує 2-гу.

Другому гравцеві зміна стратегії ніколи не змінює величину виграшу: в разі вибору 1-ої стратегії 1-им гравцем і 1-ої стратегії 3-ім гравцем, 2-ий незалежно від свого вибору отримує 1; в разі вибору 1-ої стратегії 1-им гравцем і 2-ої стратегії 3-ім гравцем, 2-ий незалежно від свого вибору отримує 3; в разі вибору 1-ої стратегії 1-им гравцем і 1-ої стратегії 3-ім гравцем, 2-ий незалежно від свого вибору отримує 0; в разі вибору 1-ої стратегії 1-им гравцем і 2-ої стратегії 3-ім гравцем, 2-ий незалежно від свого вибору отримує 2. Таким чином, формально виходить взаємне слабке домінування стратегій одна одною, хоча водночас це більш схоже на рівність (рівноцінність).

Для 3-го гравця, ніяка стратегія не домінує іншу ні строго, ні слабко, бо, наприклад, в разі вибору 1-ої стратегії 1-им гравцем і 1-ої стратегії 2-им гравцем, 3-му вигідніше вибрати свою 2-гу стратегію й отримати 3, ніж вибрати свою 1-шу стратегію й отримати 0; але, водночас, в разі вибору 2-ої стратегії 1-им гравцем і 1-ої стратегії 2-им гравцем, 3-му вигідніше вибрати свою 1-шу стратегію й отримати 2, ніж вибрати свою 2-гу стратегію й отримати 0. Тобто, в одних ситуаціях строго більший виграш дає одна стратегія, а в інших — інша.

Задача 0.0:28. «games028»

Напишіть програму, яка застосує до вказаних матриць виграшів біматричної гри двох гравців послідовне виключення строго домінованих стратегій. Тобто, поки хоча б у одного з гравців є стратегії (хоча б одна, можна більше), строго домінуванні іншими його ж стратегіями, домінуванні стратегії слід вилучати; цю дію треба виконати для обох гравців; може виявлятися (а може й не виявлятися), що, внаслідок вилучення стратегій одного з гравців, для іншого гравця деякі зі стратегій, які досі не були строго домінованими, стають такими; якщо таке відбувається, їх теж слід вилучати. Див. також задачу «Домінування стратегій біматричної гри» та примітки наприкінці цієї умови.

Вхідні дані. У першому рядку через пропуск (пробіл) задано кількість стратегій першого гравця N ($2\leqslant N\leqslant 12$) та кількість стратегій другого гравця M ($2\leqslant M\leqslant 12$). Далі йде один порожній рядок. Наступні N рядків містять платіжну матрицю виграшів першого гравця, тобто кожен з цих рядків містить рівно M чисел, розділених одинарними пропусками (пробілами), причому j-е число i-го рядка являє собою виграш першого гравця у випадку, якщо перший гравець застосує стратегію №i, а другий стратегію №j. Далі йде один порожній рядок, після якого в аналогічному форматі задано матрицю виграшів другого гравця: теж N рядків по M чисел, де j-е число i-го рядка являє собою виграш другого гравця у випадку, якщо перший гравець застосує стратегію №i, а другий стратегію №j. Всі значення елементів матриць є цілими числами, що не перевищують за модулем (абсолютною величиною) 999.

Результати. Ваша програма повинна вивести «решти» матриць виграшів, що лишаються після викреслень, у такому вигляді: спочатку «решту» матриці виграшів прешого гравця, потім дру́гого. Кожна з цих «решт» може бути утворена таким чином: взята відповідна матриця зі вхідних даних, зліва дописані номери стратегій першого гравця `a1`, `a2`, …, а згори номери стратегій другого гравця `b1`, `b2`, …, після чого відбуваються власне всі викреслення. У наведених прикладах результати матриці відформатовані так, щоб створювати матриці з акуратними стовпчиками, але при автоматичній перевірці це не перевірятиметься (досить, щоб всередині кожного з рядків відповіді утворювалася та сама послідовність позначок стратегій та чисел).

Приклади:

Клавіатура (stdin)	Екран (stdout)
3 2 1 2 4 8 6 4 11 12 21 22 31 32	b2 a2 8 b2 a2 22
3 2 1 2 4 8 1 2 11 12 91 22 31 32	b1 a2 4 b1 a2 91
3 2 1 2 4 8 6 4 11 12 29 28 31 32	b1 b2 a2 4 8 a3 6 4 b1 b2 a2 29 28 a3 31 32

Примітка. Перший тест передбачає, наприклад, такі перетворення:

(Тут і далі, в кожній комірці, перше число — виграш першого гравця, друге — виграш другого. Слово «наприклад» вжите в тому смислі, що існують також інші порядки викреслювань, які дають такий самий результат.)

Другий тест передбачає, наприклад, такі перетворення:

Перше викреслення правильне, бо можна, щоб і `a1`, і `a3` одночасно були строго домінованими; тут неважливо, що вони (для 1-го гравця) однакові, досить і того, що кожну з них строго домінує `a2`.

У третьому тесті вдається вилучити лише стратегію `a1`.

Задача 0.0:29. «games051»

Є одна купка, яка спочатку містить N паличок. Двоє грають у таку гру. Спочатку перший може забрати з купки або 1, або 2 палички. На кожному подальшому ході кожен з гравців може забрати будь-яку кількість паличок від 1 до подвоєної кількості, щойно забраної суперником (обидві межі включно). Інших варіантів ходу нема. Ходять гравці по черзі, пропускати хід не можна. Виграє той, хто забирає останню паличку (можливо, разом з деякими іншими).

Напишіть програму, яка визначатиме, хто виграє при правильній грі обох гравців. Іншими словами, хто з гравців може забезпечити собі виграш, хоч би як не грав інший.

Вхідні дані. Єдине ціле число N — початкова кількість паличок у купці ($2 \leqslant N \leqslant 1234567$).

Результати. Єдине ціле число, або `1` (якщо перший гравець може забезпечити собі виграш, як би не грав дру́гий), або `2` (якщо дру́гий, як би не грав перший).

Приклади:

Вхідні дані	Результати
2	1
3	2
4	1
5	2
1024	2

Примітка. Ці приклади частково пояснені також у прикладах до наступної задачі.

Оцінювання. 1-й блок (тести 1–5) містить тести з умови, перевіряється завжди, але безпосередньо не оцінюється. 2-й блок (тести 6–10) містить усі значення з діапазону $6 \leqslant N \leqslant 10$ і оцінюється у 30% балів; перевіряється й оцінюється завжди. 3-й блок (тести 11–20) має обмеження $11 \leqslant N \leqslant 100$ й оцінюється у 20% балів; перевіряється й оцінюється завжди. 4-й блок (тести 21–30) має обмеження $101 \leqslant N \leqslant 1234$ й оцінюється у 20% балів; перевіряється й оцінюється завжди. 5-й блок (тести 31–40) має обмеження $12345 \leqslant N \leqslant 43210$ і оцінюється у 15% балів; перевіряється й оцінюється, лише якщо успішно пройдено всі попередні блоки. 6-й блок (тести 41–50) має обмеження $123456 \leqslant N \leqslant 1234567$ і оцінюється у 15% балів; перевіряється й оцінюється, лише якщо успішно пройдено всі попередні блоки. Для кожного окремо взятого блоку, бали нараховуються, лише якщо успішно пройдено *всі* тести блоку.

Задача 0.0:30. «games052»

Гра та сама, що у попередній задачі «Палички з ходами, залежними від попереднього; хто переможе?».

Напишіть програму, яка інтерактивно гратиме за першого гравця.

На початку, один раз, Ваша програма повинна прочитати одне ціле число в окремому рядку — початкову кількість паличок N ($2 \leqslant N \leqslant 12345$).

Потім слід повторювати такий цикл:

1. Вивести єдине число в окремому рядку — свій хід, тобто кількість паличок, які вона зараз забирає з купки. На першому ході це повинно бути 1 або 2, на подальших — ціле число від 1 до подвоєної кількості, щойно забраної програмою-суперницею, причому не більше за поточну кількість паличок у купці.
2. Якщо після цього купка стає порожньою, вивести окремим рядком фразу "`I won!`" (без лапок, символ-у-символ згідно зразку) і завершити.
3. Прочитати хід програми-суперниці, тобто кількість паличок, які вона зараз забирає з купки (єдине число, в окремому рядку). Якщо Ваша програма правильно визначила, що гра не закінчилася і цей хід відбудеться, то гарантовано, що він допустимий (число є цілим від 1 до подвоєної кількості, щойно забраної Вашою програмою, і не перевищує поточну кількість паличок у купці).
4. Якщо після цього купка стає порожньою, вивести окремим рядком фразу "`You won...`" (без лапок, символ-у-символ згідно зразку) і завершити.

Це слід повторювати, доки якась із програм-гравців не ви́грає. Програма-суперниця не виводить фраз "`I won!`" / "`You won...`" чи якихось їх аналогів.

Оцінювання. Тести оцінюються кожен окремо (без блоків). У 1-му тесті $N = 3$, у 2-му $N = 5$, і ці тести не приносять балів. Решта тестів приносять однакові бали. У 20% тестів $2 \leqslant N \leqslant 25$ ($N \neq 3$, $N \neq 5$), програма-суперниця ідеальна (не робить помилок). Ще у 20%,

$100 < N \leqslant 1234$, суперниця ідеальна. Ще у 20%, $1234 < N \leqslant 12345$, суперниця ідеальна. Ще у 20%, $100 < N \leqslant 1234$, суперниці інші. Ще у 20%, $1234 < N \leqslant 12345$, суперниці інші. Ці інші програми-суперниці (їх кілька різних) роблять ходи, де гарантовано дотримані вимоги «забирати лише від 1 палички до подвоєної щойно забраної кількості» та «забирати не більше паличок, чим є у купці», але дотримуються кожна власних уявлень, як треба грати, частенько вибираючи не найкращий з допустимих ходів.

Буде оцінюватися і вміння Вашої програми виграти там, де це можливо, і вміння Вашої програми гідно, дотримуючись правил гри, програти, де виграш неможливий, і вміння Вашої програми скористатися (теж згідно правил) помилками чи іншими неадекватностями програми-суперниці, якщо такі будуть. За будь-яке порушення правил гри з боку Вашої програми, відповідний тест буде оцінено на 0 балів.

Приклади:

Вхід, суперник	Ваша програма
3 2	 1 You won...

У купці спочатку 3 палички. Ваша програма забирає одну, лишається дві; програма-суперниця забирає обидві й виграє.

Вхід, суперник	Ваша програма
3 1	 2 You won...

Спробуємо забрати не одну, а дві з трьох паличок, тобто лишити одну; програма-суперниця забирає її і теж виграє.

Вхід, суперник	Ваша програма
5 1 2	 1 1 You won...

У купці спочатку п'ять паличок. Ваша програма забирає одну, лишається чотири; програма-суперниця забирає одну, Вашій програмі дістається три палички, й вона ніяк не може виграти з вищеописаних причин.

Вхід, суперник	Ваша програма
5 3	 2 You won...

Спробуємо забрати дві з п'яти паличок (лишається три); програма-суперниця забирає всі три (має право, бо Ваша програма щойно взяла дві) й теж виграє.

Примітки. (1) Всі наведені послідовності ходів є прикладами правильної гри. Ваша програма не зобов'язана при різних запусках для однієї початкової кількості паличок робити різні ходи. Але вона має таке право. Якщо Ваша програма при різних запусках грає по-різному, система автоматичної перевірки не шукатиме ні найкращий, ні найгірший з результатів, а просто оцінюватиме перший. (2) Вводити/виводити порожні рядки не треба; додаткові вертикальні відступи у прикладах зроблені умовно, щоб краще було видно, хто коли ходить.

Задача 0.0:31. «games053»

Правила гри переважно відповідають задачі «Фішка на мінному полі—1» (включно з «фішка може рухатися або праворуч, або вниз, на будь-яку кількість клітинок у межах по́ля, не стаючи на міни й не перестрибуючи їх» та «нормальною умовою завершення», тобто «хто не може ходити — програ́є»), але спочатку маємо k фішок, які стоять на різних незамінованих клітинках. Кожен гравець на кожному своєму ході може рухати будь-яку фішку, але лише одну. Перестрибувати іншу фішку (чи кілька інших фішок) можна. Якщо різні фішки потрапляють в одну клітинку, вони негайно щезають.

Напишіть програму, яка визначить, хто ви́грає при правильній грі обох гравців, а якщо ви́грає 1-й гравець, то також знайде сукупність його «виграшних ходів» (після яких він все ще не втрачає свій виграш при правильній грі обох гравців).

Вхідні дані. Перший рядок містить два цілі числа́ N та M, розділені одним пробілом — спочатку кількість рядків, потім кількість стовпчиків. Обидва ці значення у межах від 1 до 123. Далі йдуть N рядків, що задають мінне поле. Кожен з них містить рівно по M символів `.` (позначає вільну клітинку) та/або `*` (позначає заміновану клітинку). Ці символи йдуть без роздільників, і кожен з цих N рядків містить лише ці символи та переведення рядка наприкінці.

Далі окремим рядком записане ціле число k ($1 \leqslant k \leqslant 12$) — початкова кількість фішок. Далі йдуть ще k рядків, кожен з яких містить по два цілі числа́, розділені одним пробілом — спочатку номер рядка, потім номер стовпчика початкового розміщення чергової фішки, причому рядки нумеруються від 1 до N згори донизу, стовпчики — від 1 до M зліва направо. Гарантовано, що початкові розміщення віх фішок різні, й кожна фішка розміщена в незамінованій клітинці.

Результати. Перший рядок має містити єдине ціле число, або `1` (якщо перший гравець може забезпечити собі виграш), або `2` (якщо дру́гий). Якщо відповідь з першого рядка `2`, то на цьому виведення слід припинити. А якщо відповідь з першого рядка `1`, то далі треба вивести також перелік всіх можливих перших ходів першого гравця, після яких дру́гий (при правильній грі першого) вже ніяк не зможе виграти. Цей перелік виводити в такому форматі: в один рядок через одинарні пробіли номер фішки, напрям (єдина буква “`D`”/“`R`” без лапок) та кількість клітинок, на які слід перемістити фішку. Порядок виведення «виграшних ходів» може бути довільним, але вони мусять бути згадані всі, кожен по одному разу. Номер фішки слід задавати числом від 1 до k, в порядку, як вони задані у вхідних даних.

Приклади:

Вхід	Рез-ти
2 4 .... .**. 2 1 1 2 4	1 1 D 1 1 R 2
1 1 . 1 1 1	2

Вхід	Рез-ти
5 7 ....*.. .*..... ....*. ..*.... 4 1 1 2 5 3 3 4 2	1 1 D 3 2 R 2 3 R 1

Задача 0.0:32. «games061»

Двоє грають у таку гру. Спочатку є N ($1 \leqslant N \leqslant 10^5$) паличок. На кожному ході кожен гравець може забирати або одну, або дві, або три палички, але не більше, чим їх є всього. Описані досі ходи (тотожні класичній грі Баше) будемо називати *традиційними*. Крім них, існують *спеціальні* ходи: задається кілька пар цілих чисел (a_1, b_1), (a_2, b_2), …, (a_t, b_t), які означають: якщо кількість паличок дорівнює якомусь із a_i, то гравець може замінити a_i паличок на b_i. Якщо гравець може зробити спеціальний хід, він сам вирішує, чи робити його, чи якийсь із традиційних, але зробити якийсь один хід треба. Ситуація, коли відразу кілька пар мають однакове значення a_i, можлива; в такій ситуації гравець теж сам вирішує, яким із доступних ходів (спеціальних чи традиційних) скористатися. В будь-якому разі, після кожного ходу черга ходити переходить до іншого гравця, пропускати хід не можна. (Щоправда, користуватися ходом, де $b_i = a_i$, можна, і це в деякому смислі відповідає пропуску ходу; але ж це можливо, лише якщо поточна кількість паличок якраз рівна таким b_i та a_i, для яких існує такий спеціальний хід.) Закінчується гра тоді, коли лишається 0 паличок (це може статися хоч після традиційного ходу, хоч, якщо існує пара, де $b_i = 0$, після спеціального), і той, хто походив у цю позицію, виграв, а той, кому така позиція дісталася, програв. Однак, ця гра може й не завершуватися, бо завдяки спеціальним ходам з $b_i > a_i$ кількість паличок може так ніколи й не стати 0. Такий результат кожен з гравців розцінює як гірший, чим виграш, але кращий, чим програш.

Напишіть програму, яка визначатиме, хто виграє при ідеальній грі обох гравців. Щоб відповідь не так легко було вгадати, задачу слід розв'язати, при одній і тій самій сукупності пар (a_1, b_1), (a_2, b_2), …, (a_t, b_t), для різних початкових кількостей паличок.

Вхідні дані. В першому рядку записане єдине ціле число t ($0 \leqslant t \leqslant 12345$), яке задає кількість пар, що утворюють спеціальні ходи. Кожен з подальших t рядків задає один спеціальний хід, у вигляді a_i b_i (два числá, розділені пробілом). Наступний ($(t+2)$-й) рядок містить єдине ціле число k ($1 \leqslant k \leqslant 12345$) задає кількість варіантів початкової кількості паличок, а ще наступний ($(t+3)$-й) рядок містить k натуральних чисел N_1, N_2, …, N_k (кожне в межах $1 \leqslant N_i \leqslant 10^5$) – різні початкові кількості паличок, для яких слід розв'язати задачу.

Результати. Виведіть у один рядок без пропусків рівно k великих латинських букв `W` та/або `L` та/або `D`, де `W` позначає, що при відповідній початковій кількості паличок гарантувати собі виграш може 1-й гравець, `L` – що 2-й, а `D` – що жоден з гравців не може гарантувати собі виграшу, але 1-й гравець може гарантувати, що або гра триватиме нескінче́нно довго, або якщо 2-й поступиться, то виграє 1-й.

Приклади:

Вхідні дані	Результати
2 1 1 8 8 12 1 2 3 4 5 6 7 8 9 10 11 12	WWWLWWWDDDDD

Вхідні дані
12 3 5 5 0 8 9 11 4 14 4 12 0 13 0 18 9 19 9 13 18 18 13 1 4 22 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22
Результати
WWWLWWWDDDWWWWLWWWDDDDD

Примітки. У першому прикладі додаткові ходи дозволяють не забирати палички, коли їх у купці або 1, або 8. Якщо в купці 1 паличка, її вигідніше забрати й виграти. Якщо в купці 8 паличок, будь-який зі традиційних ходів веде до програшу, й вигідніше нічого не забирати, що потім повторюється обома гравцями до нескінче́нності. Й так виходить, що з усіх подальших позицій (9, 10, … палчик) теж вигідніше якось (за один хід чи за кілька) прийти до позиції «8 паличок» й повторювати її до нескінче́нності.

У другому прикладі все набагато складніше.

Задача 0.0:33. «games071»

N карток викладені в ряд зліва направо. На кожній картці написано два натуральні числа, одне вгорі й цианового (ось такого) кольору, інше внизу й фіолетового (ось такого) кольору. Два гравці по черзі забирають по одній картці, причому забирати можна або крайню ліву, або крайню праву. Закінчується гра, коли забрано всі картки, пропускати хід не можна. Мета гри — отримати якомога більшу суму, ***враховуючи, що 1-й гравець враховує й додає лише верхні цианові числа своїх карток, 2-й гравець враховує й додає лише нижні фіолетові числа своїх карток***.

Якими будуть результати гри при правильній грі обох гравців?

Вхідні дані. У першому рядку вказано кількість карток N ($1 \leqslant N \leqslant 30$). Далі йдуть N рядків, кожен з яких містить записані через пропуск (пробіл) рівно два натуральних числа, записані на відповідній картці, спочатку верхнє цинанове, потім нижнє фіолетове. Гарантовано, що всі $2N$ чисел різні, всі є цілими степенями двійки і перебувають у межах від 2^0 до 2^{60}, обидві межі включно.

Результати. Виведіть в одному рядку через пропуск два цілі числа — результати гри при правильній грі обох гравців, спочатку суму цианових чисел, яку набере 1-й гравець, потім суму фіолетових чисел, яку набере 2-й гравець.

Приклади:

Вхідні дані	Результати
4 1024 8 256 512 4 2 16 65536	1280 65538
4 1 2 8 64 1024 256 32 16	1025 80
4 1 2 8 256 1024 64 32 16	1056 258

Задача 0.0:34. «games072»

Загальні правила гри, нарахування виграшу та формат вхідних даних такі ж, як у задачі «Гра на максимум суми (L/R, два числа на картці) — 1».

Якими будуть результати гри при правильній грі обох гравців? Яку максимальну суму гарантовано зможе набрати перший гравець?

Вхідні дані. У першому рядку вказано кількість карток N ($1 \leqslant N \leqslant 30$). Далі йдуть N рядків, кожен з яких містить записані через пропуск (пробіл) рівно два натуральних числа, записані на відповідній картці, спочатку верхнє цинанове, потім нижнє магентове. Гарантовано, що всі $2N$ чисел різні, всі є цілими степенями двійки і перебувають у межах від 2^0 до 2^{60}, обидві межі включно.

Результати. Виведіть в першому рядку через пропуск два цілі числа — результати гри при правильній грі обох гравців, спочатку суму цианових чисел, яку набере 1-й гравець, потім суму магентових чисел, яку набере 2-й гравець. Потім виведіть у дрýгому рядку одне ціле число — максимальну суму, яку гарантовано зможе набрати 1-й гравець, хоч би як не грав 2-й.

Приклади:

Вхідні дані	Результати
4 1024 8 256 512 4 2 16 65536	1280 65538 1040
4 1 2 8 64 1024 256 32 16	1025 80 1025
4 1 2 8 256 1024 64 32 16	1056 258 1025

Задача 0.0:35. «games073»

N карток викладені в ряд зліва направо. На кожній картці написане ціле число. ***Три*** гравці по черзі (строго в порядку «1-й, 2-й, 3-й, 1-й, 2-й, 3-й, …») забирають карткú, причому забирати можна лише з правого боку, або 1 картку, або 2 картки, або 3 картки (але, звісно, не більше карток, чим їх є). Закінчується гра, коли забрано всі карткú (поки хоч одна картка є, гравець зобов'язаний робити один із можливих ходів). Мета гри — отримати якнайбільшу суму (чисел, записаних на забраних картках).

Яку суму набере кожен з гравців, якщо всі вони гратимуть правильно, намагаючись лише максимізувати кожен свою суму?

Вхідні дані. У першому рядку вказано кількість карток N ($1 \leqslant N \leqslant 1234567$). У другому рядку через пробіли задані N цілих чисел (що не перевищують за модулем 10^3; інакше кажучи, з проміжку $[-1000; +1000]$) — значення, записані на картках.

Результати. Виведіть в один рядок три цілі числá, розділені пропусками — суми, які наберуть за правильної гри 1-й, 2-й і 3-й гравці відповідно.

Приклади:

Вхідні дані	Результати
7 11 11 11 11 11 11 11	33 33 11
8 3 2 999 4 6 7 -5 1	4 8 1005
9 23 -17 2 -13 5 3 11 -7 19	12 9 5

Примітки. У першому тесті, числа додатні й однакові, тому вигідно забирати якнайбільше карток: 1-й забирає три штуки, потім 2-й забирає три штуки, потім 3-й забирає останню й цим завершує гру; суми становлять $11 + 11 + 11 = 33$ для 1-го, $11 + 11 + 11 = 33$ для 2-го та 11 для 3-го.

Наведу лише структуру відповіді дрýгого тесту (не пояснюючи, чому вона така). 1-й забирає 1; 2-й забирає –5, 7 та 6; 3-й забирає 4, 999 та 2; 1-й забирає 3 й цим закінчує гру.

Задача 0.0:36. «games074»

Є a карток, на яких написано "+", та b карток, на яких написано "–"; ці написи є лише з одного боку, а з іншого боку ці картки (всі $a + b$ штук) однакові. Всі ці карткú якось випадково перемішані, й усі лежать догори тією стороною, з якої вони однакові.

Єдиний гравець може брáти ці карткú по одній, перевертати й дивитися, чи взяв "+", чи "–". За кожну картку, на якій написано "+", виграш гравця збільшується на 1, а за кожну, на якій написано "–", зменшується на 1, і може ставати від'ємним. Зокрема, якщо забрати всі карткú, то виграш буде $a - b$.

Але гравець не зобов'язаний забирати всі карткú; він має право в будь-який момент (після взяття будь-якої кáртки, вже побачивши її позначку, або на сáмому початку, ще нічого не взявши) заявити «закінчую» і припинити гру. Тоді його виграш дорівнює сумі, яку він уже набрав на той момент (включно з останньою взятою карткою, якщо така була).

Яке максимальне матсподівання виграшу може забезпечити собі гравець при правильній грі?

Вхідні дані. У єдиному рядку через пропуск (пробіл) вказано два числа a, b (обидва цілі, з проміжку від 0 до 100) — кількості карток з позначками “+” та “–” відповідно.

Результати. Виведіть єдине дійсне число — матсподівання виграшу для найкращої можливої стратегії єдиного гравця, якщо карткú перемішані випадково, а гравець знає чúсла a, b. Формат виведення може бути будь-яким зі стандартних (зокрема, байдуже, чи вивести `0.5`, чи `0.500000000`, чи `5e-1`), важливо лише забезпечити точність, вказану в «Оцінюванні».

Приклади:

Вхідні дані	Результати
2 0	2
0 2	0
7 50	0
1 1	0.5
7 9	0.299038462

Примітки. У першому прикладі, всі карткú мають позначки “+”, їх вигідно забрати всі (обидві). У дрýгому прикладі, всі карткú мають позначки “–”, їх вигідно взагалі не брáти. У третьому прикладі, мінусів настільки більше, чим плюсів, що теж вигідно взагалі не брáти. У четвертому прикладі, вигідно взяти першу картку, далі так:

- якщо на ній “$+$”, то спинитися; виграш буде 1;
- якщо на ній “–”, то взяти й наступну (останню), на ній точно буде “$+$”; виграш буде $(-1)+1=0$.

Ймовірності кожного з цих випадків $\frac{1}{2}$, тому матсподівання $\frac{1}{2}\cdot 1+\frac{1}{2}\cdot 0=\frac{1}{2}+0=\frac{1}{2}$. Також зверніть увагу, що міркування «раз плюсів і мінусів однаково, то й сума буде 0», хоч і може здатися природнім, насправді нічого не дає в цій задачі. Сáме тому, що в деяких ситуаціях видно, що вигідніше зупинитися й не забирати решту карток. П’ятий приклад надто громіздкий, щоб пояснити число-відповідь детально; але зверніть увагу, що вміння вчасно зупинятися може дати додатне матсподівання виграшу, навіть коли мінусів (трохи) більше, чим плюсів.

Оцінювання. Потестове (кожен тест перевіряють і оцінюють незалежно від решти). Тест зараховують, коли абсолютна або відносна похибка (хоча б одна з двох) не перевищує 10^{-9}.

Задача 0.0:37. «games075»

N карток викладені в ряд зліва направо. На кожній картці написане ціле число. Два гравці по черзі забирають карткú, причому забирати можна лише з правого боку, або 1 картку, або 2 картки, або 3 картки (але, звісно, не більше карток, чим їх є). Закінчується гра, коли забрано всі карткú (поки хоч одна картка є, гравець зобов’язаний робити один із можливих ходів). Крім того, після кожного ходу кожного з гравців з імовірністю 10% стається (отже, з імовірністю 90% не стається) випадкове віднесення вітром картки, яка щойно стала крайньою правою. (Це *найголовніша* відмінність цієї задачі від задачі «Гра на максимум суми (1/2/3)».) Мета гри — отримати якнайбільшу суму (чисел, записаних на забраних картках).

Які матсподівання сум, що їх наберуть гравці, якщо обидва гратимуть якнайкраще, намагаючись максимізувати кожен своє матсподівання?

Вхідні дані. У першому рядку вказано кількість карток N ($1\leqslant N\leqslant 1234$). У другому рядку через пробіли задані N цілих чисел (що не перевищують за модулем 10^3; інакше кажучи, з проміжку $[-1000;+1000]$) — значення, записані на картках.

Результати. Виведіть в одному рядку через пропуск два дійсні числá — матсподівання сум, що наберуть гравці (спочатку 1-й, потім 2-й). Формат виведення може бути будь-яким зі стандартних (зокрема, байдуже, чи вивести `41.91`, чи `41.910000000`, чи `4.191e+1`), важливо лише забезпечити точність, вказану в «Оцінюванні».

Приклади:

Вхідні дані	Результати
7 11 11 11 11 11 11 11	41.91 33
8 3 2 999 4 6 7 -5 1	810.17 116.2
9 23 -17 2 -13 5 3 11 -7 19	26.7633 2.215

Примітки. У першому тесті, числа додатні й однакові, тому гравцям вигідно забирати якнайбільше карток: спочатку 1-й забирає три штуки; потім вітер чи то відносить одну картку, чи то ні, й карток лишається чи то три штуки, чи то чотири; в будь-якому разі, 2-й забирає три штуки; залежно від вітру, повторний хід 1-го гравця може бути, а може й не бути: він буде з ймовірністю $0{,}9 \cdot 0{,}9 = 0{,}81$, якщо вітер не віднесе картки ні після ходу 1-го гравця, ні після ходу 2-го; в цьому разі, лишиться одна картка, яку вигідно забрати. Тобто, 1-й з імовірністю 81% набере $33 + 11 = 44$, а з імовірністю $100\% - 81\% = 19\%$ набере 33; матсподівання дорівнює $0{,}81 \cdot 44 + 0{,}19 \cdot 33 = 35{,}64 + 6{,}27 = 41{,}91$. Для цих вхідних даних, вітер не може завадити 2-му гравцеві взяти три картки по 11 кожна, й результат 2-го завжди $11+11+11 = 33$. (Від вітру залежить, *які* це будуть три картки; але на кожній з них однакове 11, тому на результат не впливає.)

У дру́гому прикладі, як і в задачі «Гра на максимум суми (1/2/3)» значення 999 усе ще настільки перевищує всю решту, що все ще варто дбати в першу чергу про те, щоб узяти це найбільше число. Через те, що тепер є випадкові впливи вітру, перший хід «узяти дві картки 1 та (–5)» вже не гарантує цього, але все ще дає найбільшу ймовірність $0{,}9 \cdot 0{,}9 = 0{,}81$, якщо вітер не віднесе картки ні після ходу 1-го гравця, ні після ходу 2-го.

Як від ще детальніших пояснень відповіді дру́гого тесту, так і від будь-яких коментарів щодо третього тесту утримаюся.

Оцінювання. Потестове (кожен тест перевіряють і оцінюють незалежно від решти). Тест зараховують, коли абсолютна або відносна похибка (хоча б одна з двох) не перевищує 10^{-9}.