**Мирні коні**

|  |  |
| --- | --- |
| **Ім’я файлу-розв’язка:** | peace.\* |
| **Ім’я вхідного файлу:** | peace.in |
| **Ім’я вихідного файлу:** | peace.out |
| **Обмеження по часу:** | 3000 мс |
| **Обмеження по пам’яті:** | 256 M |

Розглянемо наступну задачу. Є стандартна шахматна дошка розміром 8 на 8 клітинок. Необхідно розставити на ній *N* коней, так, щоб вони не били один одного, при умові, що на *i*-ій горизонталі повинно стояти рівно *ri* коней, а на *j*-ій вертикалі повинно стояти рівно *cj* коней, де сума всіх *ri* рівна сумі всіх *cj* і рівна *N*. Шахматний кінь - це фігура, хід якої – це переміщення на дві клітинки в горизонтальному або вертикальному напрямку, а потім ще на одну - в перпендикулярному напрямку. Вважається, що від боєм знаходяться тільки ті клітинки, в яких кінь може закінчити свій хід.   
Потрібно за заданими значеннями *ri* знайти такі значення для *cj*, щоб вищеподана задача мала рівно один розв’язок.

**Формат вхідного файлу**

В першому рядку вхідного файлу записано вісім чисел - *r*1, *r*2,..., *r*8 (0 ≤ *ri* ≤ 8).

**Формат вихідного файлу**

В вихідний файл виведіть через прогалину шукані вісім чисел - *c*1, *c*2,..., *c*8 (0 ≤ *cj* ≤ 8). Якщо неможливо потрібним чином підібрати, то виведіть через прогалину вісім чисел -1. Якщо підходящих відповідей декілька, то виведіть лексикографічно мінімальну.

**Пример**

|  |  |
| --- | --- |
| **Peace.in** | **Peace.out** |
| **3 3 3 3 3 3 3 3** | **0 8 0 0 8 0 0 8** |
| **4 4 4 4 4 4 4 4** | **-1 -1 -1 -1 -1 -1 -1 -1** |
| **2 3 4 3 2 3 4 3** | **0 2 4 2 4 4 4 4** |

**Додавання і видалення точок**

|  |  |
| --- | --- |
| **Ім’я файлу-розв’язка:** | Add\_del.\* |
| **Ім’я вхідного файлу:** | Add\_del.in |
| **Ім’я вихідного файлу:** | Add\_del.out |
| **Обмеження по часу:** | 1000 мс |
| **Обмеження по пам’яті:** | 256 M |

Не все *N*2log*N* одинаковы полезны...

Якась лекція Сергія Копеліовича…

У вас в кожен момент є мультимножина *A* точок на площині.   
Потрібно навчитися обробляти запити трьох типів:

* Добавити точку в мультимножину *A*
* Видалити точку із мультимножини *A*
* Порахувати ∑max *distance(p, q)*.  
  p∈Aq∈A

**Формат вхідних даних:**

Число запитів *N* (1 ≤ *N* ≤ 3000). Далі *N* рядків, що описують запити(див. приклад вхідних даних). Координати точок – цілі числа від 0 до 3000. Точки можуть співпадати. Запити *видалити точку* повинен видаляти рівно одну точку (гарантується, що вона на той момент в мультимножині є).

**Формат вихідних даних:**

Після виконання кожної операції потрібно вивести поточну суму максимальних відстаней.   
Абсолютна похибка не повинна перевищувати 10-6.

**Примеры:**

|  |  |
| --- | --- |
| **Add\_del.in** | **Add\_del.out** |
| **6 + 0 0 + 5 5 + 5 0 - 5 5 - 5 0 - 0 0** | **0.00000000000000000000 14.14213562373095100000 19.14213562373095100000 10.00000000000000000000 0.00000000000000000000 0.00000000000000000000** |

**Діма и машина часу**

|  |  |
| --- | --- |
| **Ім’я файлу-розв’язка:** | teleport.\* |
| **Ім’я вхідного файлу:** | teleport.in |
| **Ім’я вихідного файлу:** | teleport.out |
| **Обмеження по часу:** | 2000 мс |
| **Обмеження по пам’яті:** | 256 M |

Мама подарувала хлопчику Дімі машину часу. На жаль, ця машина працює тільки із влаштованим в неї масивом довжини *n*. Вона може привести масив до стану в будь-який момент часу. Масив в машині також не простий, а особливий. Діма може вибрати три числа — *i* и *d* (1 ≤ *j* ≤ *n*, −1000 ≤ *d* ≤ 1000), і до всіх елементів масиву з індексом *i* магічно добавиться *d*. Діма бавиться зі своїм масивом, а мама час від час задає йому запитання — яка сума всіх чисел в масиві з індексами від *f* до *t*? Діма легко справився із запитами, а чи зможете Ви?

**Формат вхідних даних:**

В першому рядку знаходяться два цілих числа *n* и *q* (1 ≤ *n, q* ≤ 105) — кількість елементів в масиві і сумарна кількість операцій і запитів.  
В наступному рядку дано *n* чисел *a1*, *a2*, . . . , *an* (−1000 ≤ *ai* ≤ 1000) — початковий стан масиву.  
В наступних *q* рядках задані операції і запити. Перший символ в рядку може бути **t**, **+** або **?**.  
Якщо рядок починається з **t**, то це опис операції з машиною часу. Тоді в рядку є ще одне число *i* (1 ≤ *i*) — номер операції чи запиту, до стану перед виконанням якої повертається масив. *i* завжди не більше номера поточної операції.  
Якщо рядок починається з  **+**, то це операція додавання. Далі слідують *i та* *d*.  
Якщо рядок починається з **?**, то це запит. Далі слідують числа *f* и *t* (1 ≤ *f, t* ≤ *n*).

**Формат вихідних даних:**

Для кожного запиту виведіть суму чисел в масиві з індексами від *f* до *t*, по одному результату в рядку.

**Приклад:**

|  |  |
| --- | --- |
| **Teleport.in** | **Teleport.out** |
| **3 5 1 2 3 ? 1 3 + 2 3 1 ? 1 3 t 1 ? 1 3** | **6 8 6** |

**І знову сума…(A4)**

|  |  |
| --- | --- |
| **Ім’я файлу-розв’язка:** | sum.\* |
| **Ім’я вхідного файлу:** | sum.in |
| **Ім’я вихідного файлу:** | sum.out |
| **Обмеження по часу:** | 3000 мс |
| **Обмеження по пам’яті:** | 256 M |

Реалізуйте структуру даних, що підтримує множину S цілих чисел, з якою можна робити такі операції:

* Add(i) — добавити елемент у множину (якщо він там уже є, то множина не змінюється)
* Sum(l,r) — вивести суму елементів множини, котрі задовольняють нерівність

l <= x <= r

**Формат вхідних даних:**

Спочатку множина S пуста. Перший рядок містить число n – кількість операцій (<=3\*10^5). Наступні n рядків містять опис операцій. Кожна операція має вигляд або «+ i» або «? l r»(sum(l,r)).  
Якщо операція «+ і» йде на початку файлу або опісля іншої операції «+», то вона задає операцію add(i). Якщо вона йде після запиту «?» і результат був y, то виконується операція add((i + y) % 10^9).  
Усі числа в операціях і запитах лежать в інтервалі від 0 до 10^9 включно.

**Формат вихідних даних:**

Для кожного запиту виведіть одне число — відповідь на запит.

**Приклад:**

|  |  |
| --- | --- |
| **Sum.in** | **Sum.out** |
| **6**  **+ 1**  **+ 3**  **+ 3**  **? 2 4**  **+ 1**  **? 2 4** | **3**  **7** |

# К-мірна часткова сума

**Вхідний файл** **k\_sum.in**

**Вихідний файл** **k\_sum.out**

**Ліміт часу:** 10 секунд

**Ліміт пам`яті:** 64 MB

***Умова***

Задано **k**-мірну таблицю чисел , де  для **j** від **1** до **k**. Для заданих знайдіть: 

***Формат вхідного файлу***

У першому рядку записано число **k** .

У другому рядку записано розміри таблиці -  ****.

Далі записано рядків по чисел, які не перевищують 1000, що описують таблицю.

У черговому рядку записано число **q** - кількість запитів. У наступних **q** рядках описано запити:

** **.

***Формат вихідного файлу***

Виведіть **q** чисел в окремих рядках – відповіді на запити.

***Приклад вхідного та вихідного файлів***

|  |  |
| --- | --- |
| **k\_sum.in** | **k\_sum.out** |
| **3**  **2 3 5**  **1 2 3 4 5**  **5 4 3 2 1**  **2 3 1 5 4**  **1 2 3 4 5**  **5 4 3 2 1**  **2 3 1 5 4**  **5**  **1 1 1 1 2 2**  **1 1 1 2 2 2**  **1 2 3 2 3 4**  **1 3 4 2 3 5**  **1 2 4 2 2 5** | **12**  **24**  **22**  **18**  **6** |

**Поливання (water)**

У фермера Джона є поле з квітами. Воно уявляє собою прямокутник розмірами MxN (1<=M<=6, 1<=N<=100). На кожній клітинці може знаходитися або кущик з квітами, або невеличкий „плацдарм” у вигляді квадратної плитки. Вам необхідно встановити поливалки для поливу усіх квітів. Поливалку можна ставити або на клітинку з квітами (тоді вона поливає лише цю клітинку), або на границі двох клітинок з квітами (тоді вона поливає ці дві клітинки), або у точці сходження чотирьох клітинок з квітами (тоді вона поливає ці чотири клітинки). Він хоче порахувати, скількома способами можна встановити поливалки, при цьому його не цікавлять варіанти, коли якісь клітинки поливаються кількома поли валками. Ваша задача – порахувати цю кількість по модулю P (2<=P<=10000).

**Вхідні дані у файлі WATER.DAT**

Перший рядок містить три цілих числа – M, N і P. Наступні M рядків містять по N чисел – 0, якщо у відповідній клітинці квіти, і 1, якщо плацдарм.

**Вихідні дані у файлі WATER.SOL**

Єдиний рядок має містити одне ціле число - шукану величину.

**Приклад:**

**WATER.DAT**

5 6 100

1 0 1 0 1 0

0 1 0 1 0 1

1 1 1 1 1 1

1 0 0 1 1 0

1 0 0 1 0 1

**WATER.SOL**

8