XXIX Всеукраїнська олімпіада з інформатики

Перший тур

1. Клуб брутальних людей

Степан є учасником відомого клубу брутальних людей. В цьому клубі ведуть рейтинг його учасників, який змінюється лише під час щорічних змагань.

Змагання складаються з *M* послідовних раундів. В кожному раунді *K* найкращих учасників (згідно з рейтингом) сідають навколо стола з цибулею. Після того як її починають різати, перший, хто проронить сльозу, переміщується на останню позицію рейтингу та раунд закінчується. Зверніть увагу, що рейтинг у кожного учасника різний.

Під час змагань ведеться протокол, в якому після кожного раунду записується позиція вибулого учасника, що була в нього на початку раунду (у кінці раунду він переміщується на останню позицію).

Степан знайшов протокол змагань цього року, але він забув свою позицію в рейтингу до початку змагань. Тим не менш він ще пам’ятає свою теперішню позицію в рейтингу. Допоможіть Степану згадати його позицію до початку змагань.

**Завдання**

Напишіть програму **club**, яка допоможе Степану визначити його позицію до початку змагань.

**Вхідні дані**

В першому рядку вхідного файла **club.dat** записано три цілих числа *N*, *K*, *M* — кількість учасників клубу, кількість учасників в одному раунді та кількість раундів відповідно. В другому рядку міститься *M* цілих чисел: *i*-те число *Ai* (1 ≤ *Ai* ≤ *K*) задає позицію вибулого учасника у відповідному раунді. В третьому рядку записано єдине ціле число *P* (1 ≤ *P* ≤ *N*) — позицію Степана після змагань.

**Вихідні дані**

В єдиному рядку файла **club.sol** виведіть єдине число — позицію Степана до початку змагань.

**Оцінювання**

Набір тестів складається з 4 блоків, для яких додатково виконуються такі умови:

1. 30 % балів: 1 ≤ *K* < *N* ≤ 1000, 1 ≤ *M* ≤ 10000 (104).
2. 20 % балів: 1 ≤ *K* ≤ 100, *K* < *N* ≤ 1000000 (106), 1 ≤ *M* ≤ 10000 (104).
3. 20 % балів: 1 ≤ *K* ≤ 100, *K* < *N* ≤ 1000000000 (109), 1 ≤ *M* ≤ 100000 (105).
4. 30 % балів: 1 ≤ *K* < *N* ≤ 1000000000 (109), 1 ≤ *M* ≤ 100000 (105).

**Приклад вхідних та вихідних даних**

|  |  |
| --- | --- |
| club.dat | club.sol |
| 6 2 3  1 2 1  5 | 3 |

**Пояснення.** На початку змагань Степан займав третю позицію. Після першого раунду учасник з першої сходинки опустився на останню, через це Степан піднявся на одну позицію вгору, а саме на другу позицію. Після другого раунду Степан опустився на останню позицію. Після третього раунду він піднявся на одну позицію вгору, а саме на п’яту позицію.

1. Watcha

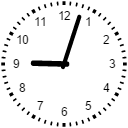
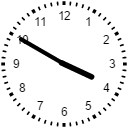
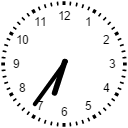
У віддаленому майбутньому, коли на Олімпії кіборги повністю замінять людей, центральна проблема, що постане перед суспільством, буде такою: тим кіборгам, які з естетичних причин надаватимуть перевагу механічному годиннику, буде важко, подивившись на годинник, розуміти, котра зараз година.

Завдання

Вирішіть цю екзистенціальну проблему вже зараз! Напишіть програму watcha, яка за виглядом механічного годинника визначатиме, котра зараз година.

Вхідні дані

Вхідний файл **watcha.dat** задає одне або кілька чорно-білих зображень годинника. Файл складається з одного або кількох блоків (щонайбільше 20 блоків у файлі). Один блок — 128 рядків по 128 символів у кожному. Кожен символ блоку — це або 0 (білий піксель: фон), або 1 (чорний піксель: елемент годинника). Між символами пропусків немає. Між собою блоки розділені порожніми рядками (по одному порожньому рядку між кожними двома сусідніми блоками). Вхідні дані завжди задають годинник одного типу, зображений нижче. Крім того, годинник завжди має однаковий розмір: перший та останній рядок і стовпець задають рамку з білих пікселів, але вже другий та передостанній рядок і стовпець містять принаймні по одному чорному пікселю годинника.

****  

Вихідні дані

Вихідний файл watcha.sol повинен містити стільки рядків, скільки блоків є у вхідному файлі. Кожен рядок задає час, відображений на годиннику у відповідному блоці (година — від 1 до 12, без нулів попереду; символ, що розділяє години та хвилини, — двокрапка).

Оцінювання

Набір складається із 50 тестів однакової вартості.

Приклад вхідних та вихідних даних

|  |  |
| --- | --- |
| watcha.dat | watcha.sol |
| *див. електронний варіант умов* | 9:03  3:50  6:36 |

1. Підчисло

Петрик та Василько — справжні друзі, тому вони постійно задають один одному всілякі цікаві задачі. Проте Василько завжди з легкістю розв’язує задачі свого друга, тож Петрик вирішив придумати по-справжньому складну задачу. І ось що в нього вийшло.

Будемо називати число *B* *підчислом* числа *A*, якщо з числа *A* можна викреслити деякі цифри так, що цифри, які залишилися, утворюють число *B*.

Задано *N*-цифрове число *X*. Позначимо як *XK* найбільше *K*-цифрове підчисло числа *X*. Необхідно відповісти на *M* запитів. Кожен запит складається з двох чисел — *K* і *L*. Відповіддю на запит є *L*-та цифра числа *XK*.

Цього разу задача справді змусила Василька задуматись. А чи зможете ви розв’язати її швидше за нього?

Завдання

Напишіть програму subnumber, яка за заданим числом та послідовністю запитів знайде необхідні цифри.

Вхідні дані

У першому рядку вхідного файла subnumber.dat міститься ціле число *X* довжини *N* (1≤*N*≤105). У другому рядку міститься число *M* (1≤*M*≤105). У наступних *M* рядках міститься по два числа *Ki*, *Li* (1≤*Ki*≤*N*,1≤*Li*≤*Ki*) — параметри *i*-го запиту.

Вихідні дані

Вихідний файл subnumber.sol повинен містити один рядок довжини *M*, в якому *i*-й символ є відповіддю на *i*-й запит.

Оцінювання

Набір тестів складається з 3 блоків, для яких додатково виконуються такі умови:

1. 15 % балів: *N* = 20, *M* = 10000.
2. 25 % балів: *N*×*M* ≤ 500000.
3. 60 % балів: *N* ≤ 100000, *M* ≤ 50000.

Приклад вхідних та вихідних даних

|  |  |
| --- | --- |
| subnumber.dat | subnumber.sol |
| 31415926  7  2 2  3 1  1 1  4 3  5 2  8 2  7 3 | 6992511 |

1. Пограбування

В країні Олімпії дуже розвинена банківська система, тому жителі залюбки кладуть свої заощадження в банк. Усього є *N* банків, які стоять в ряд. У банку номер *i* зберігається *ai* тугриків. Початково в банках немає системи безпеки, котра могла б завадити пограбуванню. Однак відомо, що якщо ввечері дня номер *d* пограбували банк номер *b*, то на ранок наступного дня необхідну систему безпеки буде встановлено у сусідніх банках (*b* − 1 та *b* + 1), після чого їх вже неможливо буде пограбувати. На ранок дня номер *d* + *i* (*i* > 0) систему безпеки буде встановлено у банках *b − i* та *b + i*. Це буде продовжуватись, поки всі банки не буде захищено від пограбування. Пограбований банк вже ніколи не можна пограбувати.

На жаль, в Олімпії навіть злочинці займаються розв’язуванням олімпіадних задач, тому уряд не сумнівається в тому, що якщо хтось замислить провести серію пограбувань, то він зробить це оптимальним чином, інакше кажучи — максимізує сумарну кількість тугриків у тих банках, які йому вдасться пограбувати до того, як в них буде встановлено систему безпеки. Крім того, відомо, що злочинці грабують не більше ніж один банк за день, а також те, що вони орудують лише ввечері.

Банківська система, аналізуючи можливі збитки від пограбувань, розглядає послідовно *M*+1 варіант розміщення коштів у банках. Кожен варіант відрізняється від попереднього кількістю грошей в одному із банків.

Завдання

Напишіть програму robbery, що для кожного з варіантів розміщення коштів у банках підрахує максимальні можливі втрати від пограбувань.

Вхідні дані

У першому рядку вхідного файла robbery.dat містяться числа *N* і *M* — кількість банків та операцій зміни кількості тугриків. У наступному рядку записано *N* чисел *ai*, що задають початкову кількість тугриків у банках. Далі йдуть *M* рядків, у кожному з яких записано пару чисел *B* та *T*, що означає, що після чергової операції в банку номер *B* стане *T* тугриків.

Вихідні дані

У вихідний файл robbery.sol для кожного *i* (0 ≤ *i* ≤ *M*) в окремий рядок виведіть максимальну кількість тугриків, яку зможуть вкрасти злочинці після виконання *i* операцій.

Оцінювання

Набір тестів складається з 3 блоків, для яких додатково виконуються такі умови:

1. 10 % балів: 1 ≤ *N*, *M* ≤ 8.
2. 20 % балів: 1 ≤ *N*, *M* ≤ 1000.
3. 70 % балів: 1 ≤ *N*, *M* ≤ 100 000.

Приклад вхідних та вихідних даних

|  |  |
| --- | --- |
| robbery.dat | robbery.sol |
| 7 4  6 7 5 6 2 2 4  6 5  7 2  7 6  4 6 | 17  18  18  19  19 |

**Пояснення.** У схемі, поданій далі, 0 позначає пограбований банк, а -1 — банк, в якому вже встановили систему безпеки.

Початковому варіанту розміщення відповідатимуть такі дії грабіжників:

6 7 5 6 2 2 4 — початковий стан;

6 7 5 0 2 2 4 — пограбували четвертий банк (6 тугриків);

6 7 -1 0 -1 2 4 — ранком наступного дня встановили систему безпеки в сусідні банки;

6 0 -1 0 -1 2 4 — пограбували другий банк (7 тугриків);

-1 0 -1 0 -1 -1 4 — систему безпеки встановлено в перший банк (через пограбування другого) та в шостий (через пограбування четвертого 2 дні тому);

-1 0 -1 0 -1 -1 0 — грабують останній банк (4 тугрики).

В сумі грабіжники мають 17 тугриків.

Останній варіант розміщення буде таким: 6 7 5 6 2 5 6. Всього грабіжники можуть отримати 6 + 7 + 6 = 19 тугриків, якщо будуть грабувати четвертий, другий, а потім сьомий банки.