# I. Backtracking and Recursive

## Bài 1: 9x9 Square

Cho một bảng hình vuông kích thước 9x9, mỗi ô của bảng là một số nguyên từ 1 tới 9 hoặc rỗng. Bảng này có thể xem là 9 hình vuông con mỗi hình là một bảng nhỏ hơn có kích thước 3x3. Hãy điền các số từ 1 tới 9 vào bảng sao cho:

1. Không hàng nào chứa hai số giống nhau
2. Không cột nào chứa hai số giống nhau
3. Không hình vuông con nào chứa hai số giống nhau

Có thể giả sử bài toán chỉ có một nghiệm duy nhất hoặc vô nghiệm, trong trường hợp có nghiệm hãy in ra nghiệm đó.

### Input

Dữ liệu của chương trình được cho trong một file text theo định dạng sau: dòng đầu là một số nguyên N (tương ứng với số test trong file), mỗi test được ghi trên 9 dòng, mỗi dòng gồm 9 số, các số từ 1 tới 9 tương ứng với các ô có số còn các số 0 tương ứng với ô trống.

### Output

Kết quả xử lý của chương trình ghi vào một file text, với mỗi test cần ghi kết qủa tìm được và số thứ tự của test.

### Ví dụ

|  |  |
| --- | --- |
| **Input** | **Output** |
| **2**  **0 0 0 9 1 0 0 0 6**  **4 0 6 8 0 5 0 0 0**  **0 2 0 4 6 0 0 7 0**  **0 4 1 0 0 0 0 0 0**  **2 0 5 0 0 0 4 0 3**  **0 0 0 0 0 0 2 5 0**  **0 1 0 0 4 8 0 3 0**  **0 0 0 3 0 1 5 0 7**  **7 0 0 0 5 6 0 0 0**  **2 0 0 0 8 5 0 0 0**  **9 5 0 0 0 6 0 7 0**  **0 0 0 0 0 0 4 3 0**  **0 6 0 0 0 0 8 0 0**  **4 0 0 8 0 1 0 0 2**  **0 0 9 0 0 0 0 4 0**  **0 7 6 0 0 0 0 0 0**  **0 2 0 1 0 0 0 9 7**  **0 0 0 2 5 0 0 0 0** | **Instance 1:**  **8 5 7 9 1 2 3 4 6**  **4 3 6 8 7 5 9 1 2**  **1 2 9 4 6 3 8 7 5**  **3 4 1 5 2 9 7 6 8**  **2 6 5 1 8 7 4 9 3**  **9 7 8 6 3 4 2 5 1**  **5 1 2 7 4 8 6 3 9**  **6 8 4 3 9 1 5 2 7**  **7 9 3 2 5 6 1 8 4**  **Instance 2:**  **No solution.** |

## Bài 2: Dãy số Anti-Prim

Cho một dãy các số nguyên liên tiếp n, n+1, n+2, …, m, một dãy số anti-prime là một hoán vị của các số nguyên trên sao cho mỗi cặp số nguyên liên tiếp tạo thành một hợp số (không phải là số nguyên tố). Chẳng hạn nếu n = 1 và m = 10 một dãy anti-prime theo định nghĩa trên là 1, 3, 5, 4, 2, 6, 9, 7, 8, 10. Đây cũng là dãy đầu tiên theo thứ tự từ điển với n = 1 và m = 10.

Chúng ta có thể mở rộng khái niệm trên bằng cách định nghĩa bậc d của một dãy anti-prim là số nguyên mà trong đó tất cả các dãy gồm 2, 3, …, d phần tử liên tiếp của dãy tạo thành một hợp số. Dãy trên có bậc là 2 nhưng không thể là bậc 3 vì 5, 4, 2 tạo thành tổng là 11 là một số nguyên tố. Dãy anti-prime bậc 3 đầu tiên là 1, 3, 5, 4, 6, 2, 10, 8, 7, 9.

### Input

Dữ liệu input của chương trình được cho trong một file text (có thể gồm nhiều test). Mỗi test gồm 3 số nguyên n, m, d được ghi trên 1 dòng trong đó (1 ≤ n< m ≤ 1000, 2 ≤ d ≤ 10). Dòng gồm 3 số 0 là kết thúc của file input.

### Output

Với mỗi test trong file input hãy đưa ra dãy anti-prim có bậc tương ứng tìm được, nếu có nhiều kết quả thì in ra dãy đầu tiên theo thứ tự từ điển còn nếu không tìm được thì in ra thông báo “Khong tim duoc day anti-prim”.

### Ví dụ

|  |  |
| --- | --- |
| **Input** | **Output** |
| **1 10 2**  **1 10 3**  **1 10 5**  **40 60 7**  **0 0 0** | **1,3,5,4,2,6,9,7,8,10**  **1,3,5,4,6,2,10,8,7,9**  **No anti-prime sequence exists.**  **40,41,43,42,44,46,45,47,48,50,55,53,52,60,56,49,51,59,58,57,54** |

## Bài 3: Dãy số (Numerical Sequence)

Một dãy số nguyên được gọi là dãy palindrome nếu như nó là một dãy đối xứng (thứ tự các phần tử từ trái qua phải giống thứ tự các phần tử từ phải qua trái). Chẳng hạn các dãy {1, 2, 1}, {15, 78, 78, 15}, {112} là các dãy palindrome nhưng các dãy {1, 2, 2}, {15, 78, 87, 15}, {112, 2, 1} không là các dãy palindrome.

Cho một dãy số nguyên (tối đa 50 phần tử, mỗi phần tử có giá trị tối đa là 10000) có thể thay hai số nguyên liên tiếp (cạnh nhau) bằng tổng của chúng. Hãy viết chương trình xác định xem số thao tác thay thế ít nhất là bao nhiêu để nhận được một dãy số palindrome.

### Input

Dữ liệu của chương trình được cho trong một file text, các số nguyên ngăn cách với nhau bằng 1 dấu phẩy.

### Output

Đưa ra số thao tác thay thế ít nhất.

### Ví dụ

|  |  |
| --- | --- |
| **Input** | **Output** |
| **15,78,78,15** | **0** |
| **1,1,1,3** | **2** |
| **15,78,87,51** | **3** |
| **3,23,21,23,42,39,63,76,13,13,13,32,12,42,26** | **8** |

## Bài 4: Truckload

Hãy tưởng tượng chúng ta có một đống các thùng hàng cần phải chuyển tới kho hàng bằng các xe tải (truck). Tuy nhiên một xe tải không thể chở quá nhiều thùng hàng nên kế hoạch chuyển đồ là chia đôi các đống hàng thành các đống nhỏ hơn cho tới khi có thể cho lên xe tải. Tất nhiên trong quá trình chia này chúng ta có thể phải chia thành hai đống không bằng nhau (chênh lệch 1 thùng hàng). Hãy viết chương trình xác định xem cần bao nhiều xe tải cần để chuyển hết các thùng hàng tới kho hàng.

### Input

Dữ liệu của chương trình được cho trong một file text, có thể gồm nhiều test. Mỗi test gồm 2 số nguyên được ghi trên 2 dòng, dòng đầu là số các thùng hàng cần chuyền (lớn hơn 1 và nhỏ hơn hoặc bằng 10000), dòng thứ hai là số thùng hàng mà một xe tải có thể chở được.

### Output

Với mỗi test cần đưa ra số các xe tải cần thiết để chuyển các thùng hàng về kho.

### Ví dụ

|  |  |
| --- | --- |
| **Input** | **Output** |
| **14**  **3** | **6** |
| **15**  **1** | **15** |
| **1024**  **5** | **256** |

## Bài 5: Path (not available)

Cho một bảng hình chữ nhật có kích thước MxN (1 ≤ M, N ≤ 50) gồm các ô nhỏ kích thước 1x1. Ban đầu một con robot xuất phát tại ô (1,1) của bảng, mỗi bước con robot chỉ có thể đi sang phải hoặc xuống ô bên dưới ngay sát nó. Hãy viết chương trình tính xem có bao nhiêu cách để con robot có thể đi đến ô đích là ô ở hàng cuối cùng bên phải (ô (M, N)).

### Input

Dữ liệu của chương trình được cho trong 1 file text, có thể có nhiều test, mỗi test gồm 2 số nguyên M, N được ghi trên 1 dòng ngăn cách với nhau bằng 1 dấu cách.

### Output

Với mỗi test kết quả xử lý ghi trên 1 dòng của file text kết quả.

### Ví dụ

(Not available)

## Bài 6: Dãy số (step sequence number) (not available)

Một dãy số nguyên được gọi là một step sequence nếu như mỗi phần tử của nó sai khác so với các phần tử cạnh nó 1 đơn vị. Chẳng hạn sau đây là các dãy số step sequence:

1 2 3 4 3

1 2 1 2 3

1 2 3 2 1

1 2 1 2 1

0 1 2 1 2

Nhưng các dãy sau không phải là các step sequence:

1 2 3 3 4

1 0 4 2 1

1 1 1 1 1

Hãy viết chương trình đếm xem dãy các số nguyên từ 0 tới N (N ≤ 20 và nhập vào từ bàn phím) có bao nhiên dãy step sequence.

### Input và output

Nhập dữ liệu từ bàn phím và ghi kết quả ra màn hình.

### Ví dụ

Với N = 4 kết quả là 3125.

## Bài 7: Bảng nhị phân

Trên một bảng hình vuông kích thước 6x6 người ta đặt các số 0 và 1 tại các ô của bảng. Khi đó chúng ta sẽ có 12 số nguyên được viết dưới dạng nhị phân 6 bit trên 6 hàng và 6 cột của bảng. Gọi các số trên các hàng lần lượt là H1, H2, .., H6 và trên các cột lần lượt là V1, V2, .., V6. Hãy viết chương trình điền các số nhị phân vào các ô của bảng sao cho các số H1, H2, .., H6, V1, V2, .., V6 là khác nhau và tuân theo một thứ tự xắp xếp tăng dần cho trước.

### Input

Dữ liệu của chương trình được cho trong một file text gồm nhiều test, mỗi test được ghi trên một dòng gồm các xâu là tên của các số của bảng nhị phân ngăn cách với nhau bằng 1 dấu cách.

### Output

Với mỗi test hãy ghi kết quả xử lý là 6 hàng của bảng nhị phân kết quả trên 6 dòng của file output.

### Ví dụ

|  |  |
| --- | --- |
| **Input** | **Output** |
| **V2 V5 H4 V6 H5 H3 V4 H1 V3 V1 H6 H2** | **101100 111011 100011 011101 011111 111010** |
| **V4 H4 H2 V1 H5 V6 H6 H3 V3 H1 V2 V5** | **111011 011110 110011 011011 101100 101111** |
| **H3 H4 V4 V3 H2 V5 V6 H6 V1 H5 H1 V2** | **110001 011110 001110 010010 101010 100010** |

## Bài 8: Lấy bài

Có một số quân bài xếp theo một hàng trên một chiếc bàn, trên mỗi quân bài người ta viết 3 chữ cái viết hoa trong bảng chữ cái tiếng Anh (từ A tới Z). Giữa hai quân bài người ta định nghĩa mức tương đồng của chúng dựa trên các chữ cái viết trên đó, chẳng hạn các quân bài “ABC” và “ACD” có độ tương đồng là 2 vì chúng có 2 chữ cái chung, các quân bài “ABC” và “DFE” có độ tương đồng là 0 vì chúng không có chữ cái nào chung.

Chúng ta có thể loại bỏ một quân bài nếu như độ tương đồng của hai quân bài liền ngay trái và phải của nó có độ tương đồng lớn hơn hoặc bằng 2, và khi 1 quân bài bị loại bỏ các quân bài hàng xóm của nó sẽ trở thành hai hàng xóm của nhau. Các quân bài ở hai đầu tất nhiên không bao giờ bị loại bỏ.

Hãy viết chương trình xác định xem chúng ta có thể loại bỏ tối đa bao nhiêu quân bài.

### Input

Dữ liệu input của chương trình được cho trong một file text gồm các chữ cái được ghi trên các quân bài, mỗi quân bài sẽ được ghi trên 1 dòng riêng biệt (số lượng các quân bài tối đa là 50, các chữ cái trên các quân bài là khác nhau).

### Output

Với mỗi test hãy đưa ra số lượng các quân bài lớn nhất có thể loại bỏ.

### Ví dụ

|  |  |
| --- | --- |
| **Input** | **Output** |
| **ABC**  **CDE**  **ABC**  **CDE**  **ABC** | **3** |
| **ABC ABC ABC** | **1** |
| **CFA EDA DBC FEA BDA BFE CFD FBE EBD CAD** | **8** |

## Bài 9: A Knight’s Journey

Cho một bàn cờ có kích thước pxq (1 ≤ p, q ≤ 26) trong đó p là số hàng của bàn cờ đánh số từ 1, q là số cột của bàn cờ đánh số bằng các ký tự hoa bắt đầu từ A. Mỗi ô của bàn cờ được mô tả theo các đánh số trên với 1 ký tự tương ứng với cột sau đó là 1 số nguyên tương ứng với chỉ số hàng Hãy tìm một hành trình xuất phát từ 1 điểm bất kỳ đi qua tất cả các ô trên bàn cờ, mỗi lần 1 ô. Cách di chuyển của quân mã trên bàn cờ giống như các di chuyển thông thường trên bàn cờ quốc tế.

### Input

File input có thể gồm nhiều test, dòng đầu tiên của file là số nguyên tương ứng với số test trong file. Mỗi test gồm 2 số nguyên tương ứng với p và q được ghi trên một dòng riêng biệt.

### Output

Với mỗi test hãy đưa ra đường đi của quân mã gồm vị trí liên tiếp trên đường đi tìm được. Nếu không tìm được đường đi kết quả là xâu “Vo nghiem”.

### Ví dụ

|  |  |
| --- | --- |
| **Input** | **Output** |
| **3**  **1 1**  **2 3**  **4 3** | **Test #1:**  **A1**  **Test #2:**  **Vo nghiem**  **Test #3:**  **A1B3C1A2B4C2A3B1C3A4B2C4** |

## Bài 10: Checker Challenge

Cho bảng kích thước 6x6 dưới đây:

**1 2 3 4 5 6**

**-------------------------**

**1 | | x | | | | |**

**-------------------------**

**2 | | | | x | | |**

**-------------------------**

**3 | | | | | | x |**

**-------------------------**

**4 | x | | | | | |**

**-------------------------**

**5 | | | x | | | |**

**-------------------------**

**6 | | | | | x | |**

**-------------------------**

Trong 6x6 = 36 ô của bảng có 6 ô được điền bởi các dấu **x** với các đặt điểm sau: mỗi hàng, mỗi cột chỉ có 1 dấu **x**, mỗi đường chéo (không chỉ tính hai đường chéo chính) không chứa nhiều hơn 1 dấu **x**. Bảng trên có thể được mô tả bởi 6 số 2, 4, 6, 1, 3, 5 với ý nghĩa là dấu **x** ở hàng 1 tương ứng với chỉ số cột bằng 2, dấu **x** ở hàng 2 tương ứng với chỉ số cột là 4 …

Cho trước kích thước bảng (2 ≤ N ≤13), hãy viết chương trình tìm các bảng thỏa mãn điều kiện: mỗi hàng, mỗi cột chỉ có một dấu x, mỗi đường chéo không có nhiều hơn 1 dấu x.

### Input

Dữ liệu của chương trình được nhập vào từ 1 file với dòng đầu là số test trong file (M <13). Sau đó là M dòng, mỗi dòng là một số nguyên tương ứng với kích thước của bảng của test đó.

### Output

Với mỗi test hãy đưa ra 3 nghiệm đầu theo cách mô tả trong ví dụ trên, sau đó đưa ra tổng số bảng tìm được. Nếu không có nghiệm nào thì kết quả là 0.

### Ví dụ

|  |  |
| --- | --- |
| **Input** | **Output** |
| **4**  **5**  **3**  **7** | **1 3 5 2 4**  **1 4 2 5 3**  **2 4 1 3 5**  **10**  **0**  **2 4 1 3**  **3 1 4 2**  **2**  **1 3 5 7 2 4 6**  **1 4 7 3 6 2 5**  **1 5 2 6 3 7 4**  **40** |

## Bài 11: The Running Man

Một tên tội phạm vừa trốn khỏi trại giam và để trốn tránh sự truy bắt của cảnh sát hắn đã trốn vào một mê cung hình vuông. Mê cung gồm các cửa, các bức tường chắn không thể đi qua và các lối đi. Mỗi cửa có thể nối giữa nhiều lối đi khác nhau. Mê cung chỉ có 1 lối vào và cũng là lối ra duy nhất.

Cảnh sát sử dụng một điều khiển từ xa có thể khóa bất cứ cửa nào trong mê cung trừ cửa thoát. Ban đầu tất cả các cửa đều mở. Vị trí của tên tội phạm được cho biết trước và chắc chắn là trên 1 con đường nào đó trong mê cung. Từ 1 vị trí trong mê cung tên trộm có thể di chuyển theo 1 trong bốn hướng: trái, phải, trên, dưới sang ô bên cạnh và tất nhiên hắn không thể di chuyển sang ô có cửa đóng hoặc tường.

Hãy giúp cảnh sát tính toán xem có tồn tại một phương án chỉ đóng 1 cửa nào đó không để nhốt tên trộm trong mê cung, và nếu có thì có bao nhiêu cửa như vậy.

### Input

Dữ liệu input của chương trình được cho trong một file text như sau: Dòng đầu là một số nguyên tương ứng với kích thước của mê cụng, dòng thứ hai là hai số nguyên tương ứng với vị trí của tên trộm (các hàng và cột của mê cung được đánh số từ 1). Tiếp theo là mô tả về mê cung: một ma trận vuông, mỗi phần tử là một ký tự hoặc W (tường), P (lối đi), D (cửa), E (lối thoát), mỗi ký tự ngăn cách với nhau bằng một dấu cách.

### Output

Nếu có thể nhốt tên trộm trong mê cung câu trả lời là “Yes”, sau đó là dòng ghi số cửa thỏa mãn yêu cầu và mô tả vị trí của các cửa này còn nếu không thể nhốt tên trộm đáp án là xâu “No”.

### Ví dụ

|  |  |
| --- | --- |
| **Input** | **Output** |
| **5**  **4 4**  **W W W W W**  **W W D P W**  **W D P W W**  **W P D P W**  **W E W W W** | **Yes**  **1**  **4 3** |

## Bài 12: Tracks

Trong một mê cung hình chữ nhật kích thước MxN có các chướng ngại vật không thể đi qua (cũng có dạng hình chữ nhật và không chồng lên nhau) và các ô có thể đi qua được. Hãy tìm số đường đi từ một điểm xuất phát tới 1 điểm kết thúc trong mê cung và đi qua tất cả các ô có thể đi qua trong mê cung, mỗi ô không đi quá 1 lần.

### Input

Dữ liệu của chương trình được cho trong một file text như sau: dòng đầu là hai số nguyên R, C (2 ≤ R, C ≤ 25) tương ứng với số hàng và số cột của mê cung (các hàng và cột của mê cung đánh số từ 1). Dòng thứ hai và thứ ba là vị trí của điểm xuất phát và kết thúc. Dòng thứ tư là một số nguyên N là số chướng ngại vật trong mê cung. Tiếp theo là N dòng mỗi dòng là mô tả về một chướng ngại vật với tọa độ đỉnh trên bên trái và đỉnh dưới bên phải của chướng ngại vật tương ứng.

### Output

Hãy đưa ra số con đường đi từ điểm xuất phát tới đích trong mê cung

### Ví dụ

|  |  |
| --- | --- |
| **Input** | **Output** |
| **4 5**  **4 1**  **3 5**  **1**  **2 2 2 3** | **3** |

## Bài 13: Campus Buildings

Trong các trường đại học luôn có các tòa nhà lớn được đặt tên bằng những cái tên khác nhau. Thông thường các tên chính xác tương đối dài nên người ta thường sử dụng các tên viết tắt chẳng hạn như SAL hay SSL … Tuy nhiên những tên viết tắt này lại rất khó hiểu và khó nhớ với những sinh viên năm thứ nhất. Qui tắt sử dụng cho các tên viết tắt này là: Một tên viết tắt có thể sẽ tương ứng với một tên đầy đủ nếu như tất cả các ký tự của tên viết tắt đều xuất hiện trong tên đầy đủ theo đúng thứ tự. Ví dụ SAL có thể là tên viết tắt của các tên đầy đủ SALvatori hoặc Student Aeospace Laboratory nhưng không thể là tên viết tắt của AngeLeS vì thứ tự của các ký tự không đúng. Hãy viết chương trình xác định xem một tên viết tắt sẽ tương với với các tên đầy đủ nào.

### Input

Dữ liệu input của chương trình được cho trong một file text, có thể có nhiều test. Dòng đầu tiên của file chứa một số nguyên K là số test trong file. Tiếp theo mỗi test bắt đầu bằng một dòng ghi 1 số nguyên N là số các tên đầy đủ có thể có của một tòa nhà, sau đó tới dòng chứa tên viết tắt của tòa nhà.

### Output

Với mỗi test cần ghi rõ số thứ tự và các xâu có thể là tên đầy đủ của tòa nhà tìm được.

### Ví dụ

|  |  |
| --- | --- |
| **Input** | **Output** |
| 2  3  studENt aeRoSpace laBoratory  salVAtori  angeles  SAL  2  SaLvatori  Science LibrarieS  SSL | Data Set 1:  studENt aeRoSpace laBoratory  salVAtori  Data Set 2: |

## Bài 14: Quyền sở hữu công ty (Controlling Companies)

Trong nền kinh tế thị trường các công ty thường có nhiều đồng sở hữu do những người này sở hữu số lượng cổ phiếu khác nhau của công ty. Chẳng hạn công ty Fozd sở hữu 12% cổ phần của công ty Mazda. Công ty A sở hữu công ty B nếu như ít nhất 1 điều kiện sau đây xảy ra:

* A = B
* Công ty A sở hữu hơn 50% cổ phiếu của công ty B
* Công ty A kiểm soát K công ty (K > 1) C1, C2, .., CK mà mỗi công ty sở hữu xi % cổ phiếu của công ty B và x1 + x2 + …xk > 50%.

Cho một danh sách các bộ 3 (i, j, p) với ý nghĩa là công ty i sở hữu p% cổ phiếu của công ty j hãy đưa ra tất cả các cặp (h, s) mà h kiểm soát s. Có nhiều nhất 100 công ty.

### Input

File dữ liệu của chương trình được ghi trong 1 file text như sau: dòng đầu tiên là một số N tương ứng với số công ty. Tiếp theo là N dòng mỗi dòng là một bộ 3 số i, j, p như đã mô tả.

### Output

Đưa ra danh sách các cặp (h, s) trong đó h kiểm soát s.

### Ví dụ

|  |  |
| --- | --- |
| **Input** | **Output** |
| **3**  **1 2 80**  **2 3 80**  **3 1 20** | **1 2**  **1 3**  **2 3** |

## Bài 15: Fuses

Các căn hộ thường sử dụng các cầu chì (fuse) 20 amp trong các cầu dao điện (fusebox). Các thiết bị điện (appliance) sẽ được đấu nối vào các cầu chì. Nếu như ácc thiết bị điện cùng nối vào một cầu chì có tổng tải vượt quá 20 amp cầu chì sẽ bị nổ. Trong các ngôi nhà lớn chúng ta sẽ cần phải sử dụng nhiều cầu chì hơn cho một số lượng lớn các thiết bị điện. Hãy tính xem với N thiết bị điện có tải điện khác nhau thì cần sử dụng tối thiểu bao nhiêu cầu chì.

### Input

Dữ liệu của chương trình được cho trong một file text như sau: Dòng đầu là số cầu chì N (không vượt quá 10). Sau đó dòng 2 là N số nguyên (mỗi số không vượt quá 10) là tải của các thiết bị điện.

### Output

Đưa ra số tối thiểu cầu chì cần sử dụng.

### Ví dụ

|  |  |
| --- | --- |
| **Input** | **Output** |
| **2**  **10 10** | **1** |
| **10**  **10 10 10 10 10 10 10 10 10 10** | **5** |
| **6**  **4 4 8 9 9 10** | **3** |

## Bài 16: Long night of Museums

Thành phố Vienna được gọi là thành phố văn hóa vì (ngoài những thứ khác nữa) có khoảng hơn 100 bảo tàng trong thành phố. Do đó rất khó (và rất đắt) để có thể đi thăm tất cả các bảo tàng trong thành phố. Thật may là có một lễ hội đêm đặc biệt gọi là “Long night of Museums”, vào đêm hội đó chúng ta có thể đi thăm bao nhiêu bảo tàng tùy thích từ 6:00 pm đến 1:00 am ngày hôm sau.

Tuy nhiên không thể đi thăm tất cả các bảo tàng vì 2 lý do chính sau: một số bảo tàng không liên quan tới sự kiện này vì sẽ đóng cửa vào 5 giờ chiều, không đủ thời gian trong 7 tiếng để đi thăm tất cả các bảo tàng.

Cho một danh sách các bảo tàng tham gia và đêm hội Long Night of Museums, thời gian đủ để đi thăm bên trong mỗi bảo tàng và thời gian để đi từ bảo tàng này sang bảo tàng khác hãy tìm một hành trình sao cho số lượng bảo tàng đi thăm được trong đêm hội là nhiều nhất.

### Input

File input có thể có nhiều test: mỗi test bắt đầu bằng một dòng chứa một số nguyên N chỉ ra số bảo tàng tham gia vào sự kiện văn hóa này (1 ≤ N ≤ 20). Mỗi bảo tàng tương ứng với một số nguyên từ 1 tới N. Dòng thứ hai của mỗi test chứa N số nguyên tương ứng với thời gian (tính bằng phút) cần thiết để đi thăm các bảo tàng từ 1 tới N. Sau đó là N dòng mô tả thời gian đi từ một bảo tàng tới các bảo tàng khác. Dòng thứ i chứa N số nguyên Mk (1 ≤ k ≤ N) biểu diễn thời gian, tính bằng phút để đi từ bảo tàng i tới bảng tàng k, số thứ i trong dòng thứ i theo mô tả này bằng 0. File input kết thúc với giá trị N = 0.

### Output

Với mỗi test cần in ra số lượng bảo tàng đi thăm lớn nhất có thể trên một dòng.

### Ví dụ

|  |  |
| --- | --- |
| **Input** | **Output** |
| **2**  **500 500**  **0 120**  **200 0**  **2**  **220 220**  **0 30**  **20 0**  **2**  **150 150**  **0 120**  **200 0**  **0** | **0**  **1**  **2** |

## Bài 17: Repeatless Numbers

Một số không lặp là một số nguyên dương mà các chữ số của nó đều là khác nhau. Chẳng hạn 25 số không lặp đầu tiên là:

1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 23, 24, 25, 26, 27, …

Cho một số nguyên dương N hãy tìm số không lặp thứ N (N ≤ 1000000), trường hợp N = 0 sẽ không xử lý

### Input

Dữ liệu input của chương trình được cho trong một file text, gồm nhiều test, mỗi test là một số nguyên dương ghi trên một dòng. Kết thúc file input là một số 0.

### Output

Với mỗi test kết quả tìm được ghi trên 1 dòng của file output.

### Ví dụ

|  |  |
| --- | --- |
| **Input** | **Output** |
| **25**  **10000**  **0** | **27**  **26057** |

## Bài 18: Transportation

Công ty Trans Inc mới đầu tư một chuyến tàu chở khách từ thành phố A tới thành phố B với M ga đỗ khác nhau. Các ga được đánh số bắt đầu từ 0 (thành phố A) cho tới M (thành phố B). Con tàu có thể chở tối đa N hành khách. Giá của vé đi tàu bằng với số ga đỗ giữa ga bắt đầu và ga kết thúc chuyến đi (bao gồm cả ga kết thúc chuyến đi). Trước khi con tàu xuất phát từ thành phố A các đơn đặt hàng vé đã được gửi về cho trung tâm thông tin của công ty xử lý. Đơn đặt hàng từ ga S có nghĩa là tất cả các vé đặt trước từ ga S tới một ga cố định nào đó. Trong trường hợp công ty không đáp ứng được tất cả các đơn đặt hàng vì hạn chế về khả năng chuyên chở thì biện pháp giải quyết sẽ là chấp nhận hoặc từ chối toàn bộ đơn đặt hàng đi từ một ga nào đó.

Hãy viết một chương trình xác định lợi nhuận lớn nhất mà công ty Trans Inc có thể thu được với một danh sách các đơn đặt hàng cho trước. Lợi nhuận thu được từ một đơn đặt hàng là tích của số hành khách nhân với giá của vé họ mua. Tổng lợi nhuận là tổng của tất cả các đơn hàng.

### Input

Dữ liệu của chương trình được cho trong một file text và có thể gồm nhiều test. Dòng đầu tiên của mỗi test là 3 số nguyên tương ứng với số người mà đoàn tàu có thể chở, số hiệu ga của thành phố B và số đơn đặt hàng từ các ga trên đường tàu chạy. Các dòng tiếp theo chứa các đơn đặt hàng. Mỗi đơn đặt hàng gồm 3 số nguyên: ga lên tàu, ga xuống tàu và số lượng hành khách. Số đơn đặt hàng tối đa là 22. Số ga trên hành trình chạy tàu tối đa là 7. Kết thúc file là một dòng ghi 3 số 0.

### Output

Với mỗi test ghi lợi nhuận lớn nhất mà công ty thu được trên 1 dòng của file output.

### Ví dụ

|  |  |
| --- | --- |
| **Input** | **Output** |
| **10 3 4**  **0 2 1**  **1 3 5**  **1 2 7**  **2 3 10**  **10 5 4**  **3 5 10**  **2 4 9**  **0 2 5**  **2 5 8**  **0 0 0** | **19**  **34** |

## Bài 19: WordGrid

Trong trò chơi đoán chữ người chơi sẽ có một bảng hình chữ nhật mà mỗi ô là một chữ cái. Họ sẽ đi tìm các từ gồm các chữ cái nằm ở các ô liên tiếp theo hàng ngang, dọc hoặc chéo (theo cả hai chiều tiến và lùi) trong bảng. Tuy nhiên một số chữ cái trong bảng đã bị mờ, hãy viết chương trình khôi phục lại các chữ cái mờ này sao cho tất cả các từ trong danh sách các từ cho trước đều có thể tìm thấy trong bảng.

### Input

Dữ liệu của chương trình được cho trong 1 file text như sau: dòng đầu của file input là 3 số nguyên, hai số đầu là kích thước của bảng trò chơi, số thứ ba là số từ xuất hiện trong bảng. Sau đó là một ma trận gồm các ký tự viết liền nhau, ký tự ‘.’ có nghĩa là ô bị mờ cần khôi phục. Và cuối cùng là danh sách các từ, mỗi từ được viết trên 1 dòng.

### Output

Hãy đưa ra bảng trò chơi sau khi đã khôi phục các ký tự bị mờ.

### Ví dụ

|  |  |
| --- | --- |
| **Input** | **Output** |
| **4 4 2**  **ABCD**  **E..H**  **I.KL**  **MNOP**  **AFK**  **DGJM** | **ABCD**  **EFGH**  **IJKL**  **MNOP** |

## Bài 20: Hotel Floor (not available)

Cho bản đồ của một tầng của khách sạn ShangriLa được mô tả bởi các ký hiệu sau:

* ‘#’ có nghĩa là tường
* ‘-‘ có nghĩa là phòng trống
* ‘\*’ có nghĩa là phòng đã có người

Hãy viết chương trình tính số người trung bình trong một phòng của khách sạn.

### Input

Dữ liệu của chương trình cho trong một file text, có thể có nhiều test, dòng đầu tiên của file input là một số nguyên tương ứng với số test trong file (ít hơn 10 test). Mỗi test được cho như sau: dòng đầu là 2 số nguyên M, N tương ứng là kích thước của 1 tầng trong khách sạn (1 ≤ M, N ≤ 100). Sau đó là M dòng mỗi dòng gồm N ký tự mô tả về các phòng trong khách sạn.

### Output

Với mỗi test ghi kết quả lên 1 dòng của file output.

### Ví dụ

|  |  |
| --- | --- |
| **Input** | **Output** |
| **2**  **5 5**  **#####**  **#\*\*##**  **###\*#**  **#\*\*##**  **#####**  **6 10**  **##########**  **#---\*--\*##**  **###-\*----#**  **#\*\*#######**  **##\*\*\*---##**  **##########** | **1.67**  **4.00** |

## Bài 21: Colored Cubes

### Input

### Output

### Ví dụ

|  |  |
| --- | --- |
| **Input** | **Output** |
| **4**  **2 4 1**  **3 1 2 4**  **1 2**  **4** | **Co the truy cap**  **1 2 4**  **Khong the truy cap**  **3** |

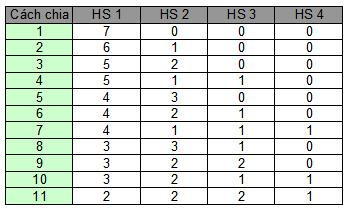
Bài 22. Chia kẹo http://laptrinh.vn/showthread.php?t=390

## Bài toán chia phần thưởng

**Cần chia hết m phần thưởng cho n học sinh sắp theo thứ tự từ giỏi trở xuống sao cho mỗi bạn nhận được phần thưởng không ít hơn phần thưởng của bạn xếp sau mình (có thể số phần thưởng = 0), 1<=m,n<= 70. Tính số cách chia phần thưởng.**  
***Cách phát biểu khác của bài toán:*** *Có* ***m phần thưởng*** *được chia cho* ***n học sinh*** *giỏi được xếp hạng thừ 1 đến n. Tính số cách chia phần thưởng sao cho thỏa các điều kiện sau:*

* *Số phần thưởng của học sinh hạng i phải lớn hơn hoặc bằng số phần thưởng của học sinh hạng j nếu j>i.*
* *Tất cả phần thưởng đều phải được thưởng hết cho học sinh.*

**Ví dụ:** Có 7 phần thưởng chia cho 4 học sinh sẽ có 11 cách chia sau:



**1. Phương pháp quy hoạch động:**

Gọi ***C[i,j]*** là số các chia i phần thưởng cho j học sinh, ta có một số nhận xét sau:

* **Có i phần thưởng mà chia cho 0 học sinh** thì có 0 cách chia (vì không thỏa điều kiện: ***Tất cả phần thưởng đều phải được thưởng hết cho học sinh****). Vậy với mọi i,* ***C[i,0]=0****.*
* **Có 0 phần thưởng mà chia cho j học sinh** thì có 1 cách chia (không ai có phần thưởng cả). *Vậy với mọi j,* ***C[0,j]=1****.*
* **Nếu số phần thưởng (i) ít hơn số học sinh (j)** thì những học sinh thứ i+1 đến j sẽ không có phần thưởng, do đó số cách ***chia i phần thưởng cho j*** người sẽ bằng số cách ***chia i phần thưởng cho i người*** . *Vậy với mọi* ***i<j, C[i,j]=C[i,i].***
* **Nếu số phần thưởng (i) nhiều hơn hoặc bằng số học sinh (j)** thì có 2 trường hợp:  
  **+ TH1**: người cuối cùng không có phần thưởng, tức là chỉ chia i phần thưởng cho j-1 người, trường hợp này số cách chia là ***C[ i ][ j-1 ].*  
  + TH2:** người cuối cùng chắc chắn có phần thưởng, khi đó ta sẽ lấy j phần thưởng chia cho j người, mỗi người sẽ có được 1 phần thưởng trước, lúc này **còn lại i-j** phần thưởng, tiếp tục lấy số còn lại này chia cho j người, trường hợp này số cách chia là ***C[i-j][j]***.  
  *Như vậy với mọi* ***i>=j, C[ i ] [ j ] = C[ i ][ j-1 ] + C[ i-j ][ j ].***

**2. Phương pháp vét cạn:**

# II. Graph

## Bài 1: Turn of lights

Dàn đèn chiếu sáng cho một sân vận động có 16 bóng đèn trên một bảng hình vuông kích thước 4x4 được điều khiển bằng một bảng điều khiển kích thước 4x4. Bảng điều khiển gồm 16 nút điều khiển được bố trí tương ứng với các bóng đèn và được đánh số từ 1 tới 16. Mỗi thao tác nhấn nút trên bảng điều khiển sẽ làm thay đổi trạng thái của bóng đèn tương ứng và 4 bóng đèn bên trái, phải, trên, dưới của nó. Thao tác nhấn nút và giữ trong 3 giây sẽ chỉ làm thay đổi trạng thái của bóng đèn tương ứng mà không ảnh hưởng tới các bóng đèn bên cạnh nó và được tính là hai thao tác.

Hãy viết chương trình tính xem cần thực hiện ít nhất bao nhiêu thao tác nhấn nút để có thể tắt được tất cả các bóng đèn.

### Input

Dữ liệu của chương trình được cho trong 1 file text, có thể gồm nhiều test. Mỗi test của chương trình gồm 4 dòng, mỗi dòng gồm 4 ký tự hoặc 0 (tương ứng với bóng đèn ở trạng thái tắt) hoặc 1 (tương ứng với bóng đèn ở trạng thái bật).

### Output

Với mỗi test hãy in ra số lượng ít nhất bao nhiêu thao tác để tắt hết các bóngd đèn.

### Ví dụ

|  |  |
| --- | --- |
| **Input** | **Output** |
| **1100**  **1000**  **0000**  **0000**  **0100**  **1110**  **0100**  **0000**  **1100**  **1000**  **0000**  **0001** | **1**  **1**  **3** |

## Bài 2: A bug’s Life

Giáo sư Hopper đang nghiên cứu về giới tính của một loài bọ khá đặc biệt mới được ngành sinh học quan tâm gần đây. Ông đặt một giả thiết là loại bọ này được chia thành hai loại có giới tính khác nhau và chúng chỉ kết đôi với các con cùng loài nhưng khác giới. Trong quá trình quan sát ông ghi lại các cặp kết đôi vào một quyển sổ dựa trên số hiệu của các con bọ được ghi trên lưng chúng. Giáo sư cũng giả thiết rằng loài bọ này không có con nào là lưỡng tính. Hãy viết chương trình dựa trên các số liệu ghi chép của giáo sư Hopper để xác định xem giả thiết của ông có đúng với kết quả ghi chép hay không.

### Input

Dữ liệu của chương trình được cho trong 1 file text. Dòng đầu của file input chứa một số nguyên là số test trong file input. Mỗi test bắt đầu bằng một dòng có 2 số nguyên là số con bọ (có thể lên tới 2000) và số lượng các lần ghép đôi ghi chép lại (có thể lên tới 1000000). Tiếp đó là các lần ghép đôi, mỗi lần được ghi trên 1 dòng với số hiệu của 2 con bọ tham gia vào lần ghép đôi đó.

### Output

Với mỗi test trong file input cần ghi kết quả gồm có 2 dòng: một dòng ghi số thứ tự của test, dòng còn lại ghi kết quả là “Khong tim thay dau hieu nghi ngo” nếu như không có bằng chứng chứng tỏ có thể có con bọ đồng tính hoặc “Co dau hieu nghi ngo” nếu như ngược lại.

### Ví dụ

|  |  |
| --- | --- |
| **Input** | **Output** |
| **2**  **3 3**  **1 2**  **2 3**  **1 3**  **4 2**  **1 2**  **3 4** | **Test 1:**  **Co dau hieu nghi ngo**  **Test 2:**  **Khong tim thay dau hieu nghi ngo** |

## Bài 3: Robot Testing

Một con robot đứng tại 1 ô trong một bảng hình vuông kích thước NxN. Robot hoạt động theo một chương trình gồm các lệnh được viết trước. Các lệnh của chương trình dành cho con robot gồm có U (lên), D (xuống), L (sang trái) và R (sang phải) 1 ô của bảng và chương trình này là lặp lại, có nghĩa là sau khi chạy xong lệnh cuối cùng của chương trình robot sẽ thực hiện lại lệnh đầu tiên. Robot chỉ dừng khi nó chạy ra ngoài bảng hoặc đã thực hiện quá 50000 lệnh.

Hãy viết chương trình xác định số lượng các lệnh trung bình mà robot phải thực hiện trước khi nó dừng lại. Có thể giả sử tất cả các vị trí của bảng đều có thể là vị trí bắt đầu của robot.

### Input

Dữ liệu của chương trình được cho trong 1 file text, có thể gồm nhiều test. Mỗi test được ghi trên 2 dòng như sau: dòng đầu là kích thước của bảng, N (1 ≤ N ≤ 1000), dòng thứ hai là một xâu tương ứng với chương trình của robot (độ dài không quá 50). Các lệnh của chương trình là các ký tự U, D, L, R được viết liền nhau.

### Output

Kết quả mỗi test lấy tới 3 chữ số sau dấu phẩy.

### Ví dụ

|  |  |
| --- | --- |
| **Input** | **Output** |
| **2**  **LURD**  **4**  **LDLDLDRRR**  **29**  **RRULDD**  **697**  **LLLLLDRRRRR** | **1.5**  **12501.0**  **3.375**  **53.236**  **3806.517** |

## Bài 4: Những tên Gangster ở Chicago

Thành phố Chicago vào những năm 1920 là một chiến trường của các tên gangster đường phố. Nếu như hai tên gangster đã gặp nhau chúng có thể trở thành bạn bè thân thiết hoặc những kẻ tử thù (mortal enemies) của nhau. Quan hệ của các tên gangster đường phố tuân theo qui luật sau:

1. Bạn của bạn cũng là bạn của mình
2. Kẻ thủ của bạn cũng là kẻ thù của mình

Hai tên gangster sẽ thuộc cùng một nhóm nếu chúng là bạn của nhau. Phòng cảnh sát Chicago đã thuê bạn viết một chương trình để xác định xem có tối đa bao nhiêu nhóm gangster khác nhau để có thể xác định mối quan hệ giữa các nhóm này nhằm nắm vững thông tin trong việc quản lý địa bàn.

### Input

Dữ liệu input của chương trình được cho trong một file text như sau: dòng đầu là một số N (2 ≤ N ≤ 1000) là số tên gangster. Các tên gangster được đánh số từ 1 tới N. Dòng thứ hai là một số nguyên M (1 ≤ M ≤ 5000) tương ứng với mối quan hệ ghi nhận được về các tên gangster. Tiếp theo là M dòng mỗi dòng gồm 1 ký tự và 2 số nguyên là số thứ tự của các tên gangster mô tả về một quan hệ ghi nhận được (ở dạng C p q) nếu C = ‘E’ có nghĩa là p, q là kẻ thù còn nếu C = ‘F’ thì p và q là bạn. Tất nhiên các ghi nhận này không có mâu thuẫn, nghĩa là hai tên gangster không thể vừa là bạn vừa là kẻ thù.

### Output

Hãy đưa ra số lượng lớn nhất các nhóm gangster trong thành phố.

### Ví dụ

|  |  |
| --- | --- |
| **Input** | **Output** |
| **6**  **4**  **E 1 4**  **F 3 5**  **F 4 6**  **E 1 2** | **3** |

## Bài 5: Asteroids

Hãy tưởng tượng ban đang ở một vị trí trong vũ trụ và muốn tìm đường về nhà. Xung quanh bạn là các tiểu hành tinh (Asteroids) và bạn không muốn va vào chúng. Hãy viết chương trình tìm đường đi ngắn nhất từ 1 điểm tới nhà của bạn.

### Input

Dữ liệu input của chương trình được cho trong một file text có thể lên tới 100 test. Mỗi test được cho theo định dạng sau:

Dữ liệu của mỗi test gồm 5 phần:

1. Một dòng bắt đầu có dạng “START N” trong đó 1 ≤ N ≤ 10.
2. Tiếp đến là N lát cắt ngang mô tả vũ trụ. Mỗi lát cắt là một ma trận NxN mô tả một lát cắt ngang của vũ trụ. Mỗi phần tử của ma trận hoặc là ký tự ‘O’ (ô trống) hoặc là ký tự ‘X’ (một tiểu hành tinh).
3. Vị trí bắt đầu gồm 3 số nguyên ngăn cách nhau bằng 1 dấu cách là vị trí ban đẩu của bạn trong vũ trụ.
4. Vị trí kết thúc gồm 3 số nguyên ngăn cách nhau bằng 1 dấu cách là vị trí nhà của bạn.
5. Dòng kết thúc với môt xâu “END”

Các vị trí cột trái, hàng trên cùng, và lát cắt đầu tiên được đánh số từ 0.

### Output

Với mỗi test kết quả cần đưa ra gồm 2 số X, Y ghi trên 1 dòng nếu có thể tìm ra đường đi trong đó X là số N tương ứng với test còn Y là số thao tác di chuyển ít nhất để về tới nhà. Ngược lại nếu không thể tìm ra đường về nhà hãy in kết quả là xâu “Khong co duong di”. Tại mỗi bước chúng ta có thể di chuyển sang ô ngay bên trái, phải, lên xuống, trước sau (6 khả năng).

### Ví dụ

|  |  |
| --- | --- |
| **Input** | **Output** |
| **START 1**  **O**  **0 0 0**  **0 0 0**  **END**  **START 3**  **XXX**  **XXX**  **XXX**  **OOO**  **OOO**  **OOO**  **XXX**  **XXX**  **XXX**  **0 0 1**  **2 2 1**  **END**  **START 5**  **OOOOO**  **OOOOO**  **OOOOO**  **OOOOO**  **OOOOO**  **OOOOO**  **OOOOO**  **OOOOO**  **OOOOO**  **OOOOO**  **XXXXX**  **XXXXX**  **XXXXX**  **XXXXX**  **XXXXX**  **OOOOO**  **OOOOO**  **OOOOO**  **OOOOO**  **OOOOO**  **OOOOO**  **OOOOO**  **OOOOO**  **OOOOO**  **OOOOO**  **0 0 0**  **4 4 4** | **1 0**  **3 4**  **Khong co duong di** |

## Bài 6: Servicing stations

Một công ty buôn bán máy tính cung cấp máy tính cho N thị trấn (3 ≤ N ≤ 35). Các thị trấn được đánh số từ 1 tới N. Có các con đường trực tiếp nối giữa M cặp thị trấn trong số N thị trấn này. Công ty quyết định mở một số trạm bảo hành tại một số thị trấn để sao cho bất cứ một thị trấn nào cũng có một trạm bảo hành hoặc tại một thị trấn nào đó ngay cạnh nó (có đường đi trực tiếp tới thị trấn cạnh đó).

Hãy viết một chương trình giúp công ty xác định xem số lượng tối thiểu các trạm bảo hành cần xây dựng là bao nhiêu.

### Input

Dữ liệu của chương trình được cho trong một file text như sau: Dòng đầu của file là 2 số nguyên N, M. Tiếp theo là M dòng mỗi dòng gồm 2 số nguyên là số thứ tự của hai thị trấn có đường đi trực tiếp.

### Output

Kết quả xử lý của chương trình cần đưa ra là số nhỏ nhất các trạm bảo hành cần xây dựng.

### Ví dụ

|  |  |
| --- | --- |
| **Input** | **Output** |
| **8 12 1 2 1 6 1 8 2 3 2 6 3 4 3 5 4 5 4 7 5 6 6 7 6 8** | **2** |

## Bài 7: Web

Trên World Wild Web một trang web có thể chứa các siêu liên kết tới các trang khác. Một quản trị viên web muốn xác định các trang có thể truy cập tới qua việc truy cập vào trang chủ cũng như các trang không thể truy cập tới từ trang chủ. Hãy viết chương trình giúp anh ta thực hiện việc này.

### Input

Dòng đầu tiên của file input là một số nguyên N là số trang web. Sau đó là N nhóm dữ liệu mô tả về các siêu liên kết của các trang web này. Mỗi nhóm mô tả bắt đầu bằng một số nguyên từ 1 tới N tương ứng với một trang web, sau đó là danh sách các trang có thể truy cập tới từ trang này (N ≤ 20). Chú ý là chúng ta có thể chuyển tới trang X từ trang Y nhưng ngược lại thì chưa chắc.

### Output

Giả sử trang chủ là trang số 1. Hãy xác định các trang không thể truy cập và có thể truy cập từ trang chủ.

### Ví dụ

|  |  |
| --- | --- |
| **Input** | **Output** |
| **4**  **2 4 1**  **3 1 2 4**  **1 2**  **4** | **Co the truy cap**  **1 2 4**  **Khong the truy cap**  **3** |

## Bài 8: Critical Network Lines

Một mạng truyền thông gồm một tập các nút mạng và các đường nối hai chiều trực tiếp giữa hai nút mạng. Một số nút của mạng cung cấp dịch vụ loại A cho tất cả các nút mạng khác (bao gồm cả bản thân nó), trong khi một số nút khác lại cung cấp dịch vụ loại B cho tất cả các nút khác (bao gồm cả bản thân nó). Một nút cũng có thể cung cấp cả hai loại dịch vụ. Tất cả các nút đều phải có khả năng truy cập vào cả hai loại dịch vụ này.

Nếu như một đường kết nối trực tiếp nào đó bị ngắt khỏi mạng thì một nút nào đó trong mạng có thể không truy cập được vào một dịch vụ trong mạng, các đường kết nối đó được gọi là các đường kết nối trực tiếp quan trọng và cần phải có kế hoạch xây dựng đường dự trữ cũng như các biện pháp bảo đảm an toàn.

Hãy viết một chương trình xác định số lượng các đường kết nối quan trọng và các điểm đầu nút của các đường này biết rằng luôn có một đường truyền (gián tiếp hoặc trực tiếp) giữa hai nút mạng bất kỳ.

### Input

Dòng đầu tiên của file input chứa 4 số nguyên N, M, K và L (1 ≤ N ≤ 1000). N là số lượng các nút trong mạng, M (1 ≤ M ≤ 10000) là số các đường nối trực tiếp trong mạng, K (1 ≤ K ≤ N) là số các nút mạng cung cấp dịch vụ A và L (1 ≤ L ≤ N) là số các nút cung cấp dịch vụ B. Dòng thứ hai của file input chứa K số nguyên là các nút mạng cung cấp dịch vụ A, dòng thứ 3 chứa L số nguyên là các nút cung cấp dịch vụ B. Tiếp theo là M dòng, mỗi dòng gồm 2 số nguyên p, q (1 ≤ p, q ≤ N, p ≠ q) là các đường nối trực tiếp trong mạng. Có nhiều nhất một đường nối trực tiếp giữa 2 nút bất kỳ trong mạng.

### Output

Dòng đầu tiên của file output chứa số lượng các đường nối trực tiếp quan trọng S, sau đó là S dòng, mỗi dòng gồm 2 số nguyên mô tả các đường nối đó.

### Ví dụ

|  |  |
| --- | --- |
| **Input** | **Output** |
| **9 10 3 4**  **2 4 5**  **4 9 8 3**  **1 2**  **4 1**  **2 3**  **4 2**  **1 5**  **5 6**  **6 7**  **6 8**  **7 9**  **8 7** | **3**  **3 2**  **5 6**  **7 9** |

## Bài 9: Doorman

Mr Magi là một quản gia (butler) trong một tòa nhà (mansion: tòa nhà lớn, lâu đài) lớn. Tòa nhà này có rất nhiều phòng và được đánh số bằng các số nguyên từ 0. Ông chủ của Mr Magi là một người rất đãng trí và thường xuyên không đóng cửa của tất cả các căn phòng trong một tầng cụ thể nào đó của khu nhà. Trải qua nhiều năm làm việc Mr Magi đã quen với đi qua các phòng bị mở cửa theo một con đường đi duy nhất trong tòa nhà và đóng các cửa sau khi đi qua. Hãy viết một chương trình giúp Mr Magi xác định xem có tồn tại một con đường đi qua tất cả các phòng bị mở cửa để đóng chúng lại thỏa mãn:

1. Luôn đóng cửa sau khi đi qua phòng đó
2. Không bao giờ mở các cửa bị đóng
3. Kết thúc công việc với tất cả các cửa đã đóng phòng làm việc của Mr Magi.

### Input

Dữ liệu của chương trình được cho trong một file text, có thể gồm nhiều test (tối đa là 100 test). Mỗi bộ test gồm 3 phần được cho theo định dạng sau:

1. Dòng đầu tiên có dạng “START M N” trong đó M là số hiệu phòng của Mr Magi và N là số các phòng trong tòa nhà (1 ≤ N ≤ 20).
2. Danh sách các phòng được cho trên N dòng. Mỗi dòng là mô tả các cửa mở của phòng đó thông sang các phòng có số hiệu lớn hơn. Chẳng hạn phòng số 3 có các cửa mở thông sang các phòng 1, 5, và 7 thì mô tả về phòng số 3 sẽ là một dòng gồm 2 số “5 7”. Dòng đầu tiên trong phần này là mô tả cho phòng số 0, dòng thứ 2 là mô tả cho phòng số 1, vân vân. Một dòng mô tả có thể là dòng trống nếu như nó không có cửa mở chẳng hạn như dòng cuối cùng là một dòng trống vì nó là mô tả cho phòng cuối cùng. Các mô tả đều tuân theo thứ tự tăng dần của số hiệu các phòng. Một phòng cũng có thể nối với nhiều phòng khác.
3. Dòng cuối cùng của mỗi test là một dòng có 1 xâu “END”.

Sau test cuối cùng là dòng có ghi xâu “ENDOFINPUT”. Chú ý là số các cửa trong mỗi test ít hơn 100.

### Output

Nếu có tồn tại một con đường như vậy hãy đưa ra thông báo kết quả là “Yes X” trong đó X là số các cửa sẽ đóng lại trên đường đi còn nếu không hãy đưa ra thông báo là “No”.

### Ví dụ

|  |  |
| --- | --- |
| **Input** | **Output** |
| **START 1 2**  **1**  **END**  **START 0 5**  **1 2 2 3 3 4 4**  **END**  **START 0 10**  **1 9**  **23456789**  **END**  **ENDOFINPUT** | **YES 1**  **NO**  **YES 10** |

## Bài 10: Walkabout

Trên một bảng kích thước NxN tại các ô có các ký tự + hoặc -. Các hàng được đánh số từ 1 tới N và từ đáy của bảng tới hàng đầu tiên, các cột được đánh số từ 1 tới N từ trái qua phải. Biết rằng tại mỗi bước đi chuyển có thể đi qua một trong các ô bên trái, phải, trên dưới của ô hiện tại và không thể đi qua các ô chứa ký tự -, hãy viết chương trình xác định xem có tồn tại một đường đi từ bên trái của bảng qua bên phải hay không.

### Input

File input có thể có nhiều test, mỗi test bắt đầu bằng một dòng ghi kích thước của bảng (là một số nguyên) sau đó là các ô tương ứng với các hàng và cột như hình vẽ sau của bảng:

(1, n) (2, n) . . . (n, n)

...

...

...

(1, 2) (2, 2) . . . (n, 2)

(1, 1) (2, 1) . . . (n, 1)

### Output

Nếu tồn tại con đường đi hãy in ra tọa độ các ô trên đường đi đó theo thứ tự, còn nếu không hãy in ra thông báo là “Không tồn tại đường đi”.

### Ví dụ

|  |  |
| --- | --- |
| **Input** | **Output** |
| **5**  **+-+++**  **--+--**  **+-++-**  **++-+-**  **-+++-**  **6**  **++++-+**  **+-+--+**  **-+-+++**  **-+++-+**  **-+-+-+**  **-+++-+** | **Mot duong di:**  **(1,2)**  **(2,2)**  **(2,1)**  **(3,1)**  **(4,1)**  **(4,2)**  **(4,3)**  **(3,3)**  **(3,4)**  **(3,5)**  **(4,5)**  **(5,5)**  **Khong ton tai duong di** |

## Bài 11: Sói và cừu (Koze)

Mr Mickey có nuôi một đàn cừu trong sân sau nhà mình Backyard (sân sau). Trong khi Mr Mickey đang ngủ say các con sói đói ăn đã vào và tấn công các chú cừu. Sân sau nhà Mr Mickey là một hình chữ nhật gồm các ô hình vuông được mô tả như sau: nếu ô chứa ký tự “.” thì đó là một ô trống, ‘#’ có nghĩa là một bức rào chắn, ‘k’ có nghĩa là một con cừa và ‘v’ là một con chó sói.

Chúng ta gọi 2 ô là cùng thuộc về một vùng nếu như chúng ta có thể di chuyển từ một ô này sang 1 ô khác bằng một con đường gồm các thao tác dịch chuyển ngang, dọc trên các ô không có các rào ngăn cách. Các ô mà từ đó chúng ta có thể di chuyển ra bên ngoài sân không được xem là một phần của bất cứ vùng nào.

Thật may mắn là các chú cừu đã được Mr Mickey huấn luyện võ Karate và có thể chiến đấu và đánh thắng bọn sói trong một vùng nếu như số lượng cừu trong vùng đó đông hơn số lượng sói. Ngược lại sói sẽ ăn thịt tất cả các chú cừu trong vùng nó chiếm số lượng đông hơn.

Ban đầu tất cả các chú cừu và sói đều nằm trong các vùng của sân.

Hãy viết chương trình đếm xem số lượng cừu và sói còn sống vào buổi sáng ngày hôm sau.

### Input

Dòng đầu tiên của file input chứa hai số nguyên R và C (3 ≤ R, C ≤ 250) là số hàng và cột của sân nhà Mr Mickey.

Tiếp đến là R dòng, mỗi dòng chứa C ký tự mô tả sân.

### Output

Đưa ra số lượng cừu và sói còn sống vào buổi sáng ngày hôm sau.

### Ví dụ

|  |  |
| --- | --- |
| **Input** | **Output** |
| **6 6**  **...#..**  **.##v#.**  **#v.#.#**  **#.k#.#**  **.###.#**  **...###** | **0 2** |
| **8 8**  **.######.**  **#..k...#**  **#.####.#**  **#.#v.#.#**  **#.#.k#k#**  **#k.##..#**  **#.v..v.#**  **.######.** | **3 1** |
| **9 12**  **.###.#####..**  **#.kk#...#v#.**  **#..k#.#.#.#.**  **#..##k#...#.**  **#.#v#k###.#.**  **#..#v#....#.**  **#...v#v####.**  **.####.#vv.k#**  **.......####.** | **3 5** |

## Bài 12: Heavy Transportation

Cho bản đồ của thành phố gồm các con đường (và trọng tải cho phép) giữa các điểm ngã ba, ngã tư, vân vân được đánh số từ 1 tới N. Hãy tìm tải trọng lớn nhất có thể vận chuyển từ ngã rẽ 1 tới ngã rẽ N. Biểt rằng hệ thống đường của thành phố là hai chiều và có ít nhất 1 con đường đi từ ngã rẽ 1 tới ngã rẽ N.

### Input

Dòng đầu tiên của file input chứa số test trong file. Mỗi test bắt đầu bằng một dòng ghi 2 số N (1 ≤ N ≤ 1000) và M (là số con đường nối giữa các ngã rẽ). Tiếp theo là m dòng, mỗi dòng chứa 3 số nguyên là số hiệu hai ngã rẽ và tải trọng cho phép của con đường nối giữa chúng (tải trọng là các số nguyên nhỏ hơn 10000). Có nhiều nhất 1 con phố trực tiếp nối giữa hai ngã rẽ bất kỳ.

### Output

Với mỗi test cần in ra số hiệu test (bắt đầu là 1) sau đó in ra tải trọng lớn nhất có thể có.

### Ví dụ

|  |  |
| --- | --- |
| **Input** | **Output** |
| **1**  **3 3**  **1 2 3**  **1 3 4**  **2 3 5** | **1**  **4** |

## Bài 13: System Failure

Cho một hệ thống gồm một mảng hình chữ nhật các phần tử. Một số phần tử có thể đã hỏng và hệ thống có tính dư thừa. Hệ thống vẫn hoạt động tốt miễn là nó có ít nhất n phần tử vẫn hoạt động tốt và được kết nối với nhau. Hai phần tử được xem là kết nối với nhau nếu như giữa chúng có một đường đi qua các phần tử liên tiếp ở cạnh nhau (theo chiều ngang, dọc hoặc chéo). Một tập các phần tử được gọi là có kết nối với nhau nếu như hai phần tử bất kỳ trong chúng là có kết nối với nhau.

Chúng ta cần xác định các phần tử quan trọng của hệ thống mà nếu nó hỏng sẽ làm cho hệ thống không thể hoạt động được nữa để có kế hoạch dự phòng, thay thế. Hãy viết chương trình đếm xem hệ thống có bao nhiêu phần tử quan trọng, trong trường hợp hệ thống đã không hoạt động được từ đầu thì kết quả là -1.

### Input

Dữ liệu của chương trình được cho trong một file text như sau: dòng đầu số N tương ứng với số phần tử còn hoạt động (1 ≤ N ≤ 1000) để đảm bảo hệ thống làm việc được. Sau đó là mô tả hệ thống gồm 1 ma trận các phần tử 0, 1 trong đó 1 tương ứng với phần tử còn làm việc được và 0 tương ứng với phần tử đã hỏng của hệ thống (kích thước của ma trận nhỏ hơn 50).

### Output

Với mỗi test hãy đưa ra số phần tử quan trọng của hệ thống.

### Ví dụ

|  |  |
| --- | --- |
| **Input** | **Output** |
| **4**  **00000**  **00000**  **11111** | **3** |
| **6**  **01010**  **10001**  **11011**  **10001**  **01010** | **0** |
| **2**  **101** | **-1** |

## Bài 14: All Disc Considered

Các hệ điều hành là các phần mềm lớn gồm rất nhiều gói phần mềm khác nhau và thường được phân phối trên các phương tiện lưu trữ khác nhau. Chúng ta có thể nhớ lại thời kỳ mà các hệ điều hành được lưu trữ và bán cho người dùng trên 12 đĩa mềm trong những năm đầu thập kỷ 90 hoặc 6 đĩa CD trong những năm cuối thập kỷ 90 của thế kỷ trước. Ngày nay hầu hết các hệ điều hành đều được phân phối trên các đĩa DVD, mỗi đĩa có hàng ngàn package khác nhau.

Việc cài đặt một gói phần mềm cụ thể nào đó thường đỏi hỏi một số gói khác phải được cài trước đó. Vì thế nếu như các gói phần mềm được phân phối trên các phương tiện lưu trữ một cách không thích hợp thì việc cài đặt sẽ đòi hỏi chúng ta thay đổi các đĩa cài đặt trong quá trình cài đặt do hệ thống chỉ có 1 ổ đĩa DVD. Vì thế chúng ta cần phải tiến hành quá trình cài đặt theo cách thức nào đó để quá trình thay đổi đĩa này diễn ra với số lần ít nhất.

Cho một danh sách các đĩa cài đặt và sự phụ thuộc giữa các gói phần mềm hãy tính số thao tác thay đĩa ít nhất để có thể cài đặt tất cả các gói phần mềm, giả sử hệ điều hành được lưu trên đúng 2 đĩa DVD.

### Input

File input có thể chứa nhiều test. Mỗi test bắt đầu với 3 số nguyên N1, N2, D (1 ≤ N1, N2 ≤ 50000, 0 ≤ D ≤ 100000). File DVD thứ nhất chứa N1 gói phần mềm được đánh số từ 1 tới N1. Đĩa DVD thứ hai chứa đúng N2 gói được đánh số từ N1 + 1 tới N1 + N2. Sau đó là D mô tả về sự phụ thuộc giữa các gói phần mềm này, mỗi sự phụ thuộc gồm 2 số nguyên xi và yi (1 ≤ xi, yi ≤ N1 + N2) với ý nghĩa là gói phần mềm xi đỏi hỏi phải có gói phần mềm yi cài đặt trước đó. Kết thúc file input là một dòng ghi 3 số 0.

### Output

Với mỗi test cần đưa ra số thao tác thay đổi đĩa ít nhất để cài đặt tất cả các phần mềm. Ban đầu ổ đĩa DVD là trống nên thao tác đưa đĩa vào sẽ được tính là 1 thao tác và cuối cùng sau khi cài xong cần phải bỏ đĩa ra và đây cũng được tính là 1 thao tác.

### Ví dụ

|  |  |
| --- | --- |
| **Input** | **Output** |
| **3 2 1**  **1 2**  **2 2 2**  **1 3**  **4 2**  **2 1 1**  **1 3**  **0 0 0** | **3**  **4**  **3** |

## Bài 15: CrazySwitch

Hãy tưởng tượng bạn đang ở trong một ngôi nhà mà đèn chiếu sáng tại mỗi phòng được điều khiển bởi một công tắc bật có thể đặt ở một phòng khác. Ban đầu đèn trong phòng đầu tiên ở trạng thái bật và đèn ở tất cả các phòng khác đều tắt.

Giả sử bạn đang ở phòng đầu tiên và mục đích của bạn là đi đến phòng cuối cùng với đèn ở tất cả các phòng đều tắt trừ phòng cuối cùng. Bạn có thể di chuyển trực tiếp từ 1 phòng bất kỳ sang 1 phòng bất kỳ khác và có thể bật tắt bất cứ công tắc nào trong căn phòng bạn đang ở. Tuy nhiên bạn không được đi vào một phòng tắt đèn hoặc tắt đèn ở phòng hiện tại.

### Input

Các phòng và các công tắc bật đèn tương ứng được đánh số từ 0. Dữ liệu input của chương trình được cho dưới dạng một mảng các số mô tả vị trí của các công tắc bật đèn, công tắc của phòng thứ i được đặt tại phòng có số thứ tự bằng phần tử thứ i của mảng input. Số phòng của ngôi nhà nhỏ hơn 17.

### Output

Hãy đưa ra số lần di chuyển ít nhất để hoàn thành công việc. Nếu không thể thì kết quả bằng -1. Mỗi thao tác di chuyển từ phòng này sang phòng khác được tính là 1 thao tác (thao tác bật đèn không tính).

### Ví dụ

|  |  |
| --- | --- |
| **Input** | **Output** |
| **1 0** | **1** |
| **0 1** | **-1** |
| **1 2 0** | **3** |
| **4 4 3 0 5 2** | **7** |
| **7 11 1 12 6 3 0 2 6 0 0 5 9** | **15** |

## Bài 16: Mother's Milk

John có ba bình sữa có dung tích là A, B và C lít (A, B, C ≤ 20). Ban đầu các bình A, B là rỗng và bình C chứa đầy sữa. John có thể đổ sữa từ bình này sang bình khác, thao tác này thực hiện cho tới khi bình thứ nhất (bình đổ sang bình khác) hết sữa hoặc bình thứ hai (chứa sữa) đầy. Hãy viết chương trình xác định lượng sữa có thể còn lại trong bình C sau khi John đã thực hiện một số thao tác.

### Input

Dữ liệu input của chương trình gồm 3 số nguyên được ghi trên 1 dòng.

### Output

Dung tích sữa có thể có ở bình C.

### Ví dụ

|  |  |
| --- | --- |
| **Input** | **Output** |
| **8 9 10** | **1 2 8 9 10** |
| **2 5 10** | **5 6 7 8 9 10** |

## Bài 17: Agri-Net

Anh nông dân John đã trúng cử trong cuộc bầu cử ở địa phương và trở thành thị trưởng của thành phố. Một trong những lời hứa khi vận động tranh cử của John là sẽ lắp đặt đường truyền Internet để kết nối tất cả các trang trại trong phạm vi thành phố. Để thực hiện lời hứa này John cần sự giúp đỡ của bạn.

John đã lắp đặt một đường truyền Internet tốc độ cao cho trang trại của mình và muốn chia sẻ với các trang trại khác. Để giảm thiểu chi phí anh ta muốn sử dụng tối thiểu số cáp quang cần thiết để kết nối với tất cả các trang trại khác.

Cho một danh sách số cáp quang cần thiết để kết nối giữa các cặp trang trại hãy tìm độ dài cáp quang nhỏ nhất để kết nối tất cả các trang trại với nhau.

### Input

Dữ liệu của chương trình được cho trong một file text như sau: dòng đầu là số trang trại N (3 ≤ N ≤ 100). Sau đó là N dòng, mỗi dòng gồm N số nguyên ngăn cách với nhau bằng một dấu cách là ma trận độ dài cáp quang tương ứng cần thiết để nối giữa các cặp trang trại.

### Output

Ghi kết quả tìm được vào file text.

### Ví dụ

|  |  |
| --- | --- |
| **Input** | **Output** |
| **4**  **0 4 9 21**  **4 0 8 17**  **9 8 0 16**  **21 17 16 0** | **28** |

## Bài 18: Depot (Kho chứa hàng)

Một công ty có N cửa hàng bán M loại hàng hóa khác nhau trong mỗi cửa hàng. Công ty có một kho hàng lớn tại đó công việc đóng gói hàng hóa sẽ được thực hiện trước khi được chuyển tới các cửa hàng. Mỗi cửa hàng nhận các loại hàng với số lượng như nhau. Vì thế công ty có thể đóng gói một lượng cụ thể một loại sản phẩm nào đó vào một container và gán nhãn cho container đó bằng số thứ tự của sản phẩm. Các sản phẩm được đánh số từ 1 tới M. Do đó khi đóng gói xong chúng ta sẽ có N\*M container trong kho và có đúng N container được đánh số giống nhau bằng số thứ tự của một loại sản phẩm. Vì kho hàng là một tòa nhà khá hẹp nên các container được xếp theo một hàng duy nhất. Đê tăng tốc độ quá trình phân phối hàng quản lý của nhà kho muốn sắp xếp lại các container. Vì các sản phẩm được chuyển tới cho các cửa hàng bằng cách sử dụng chỉ 1 xe tải cho một cửa hàng nên mỗi xe tải sẽ chuyên chở một container của một sản phẩm và một cách sắp xếp hợp lý là như sau: M container đầu tiên sẽ chứa N loại hàng khác nhau, M container tiếp theo sẽ chứa N loại hàng khác nhau, vân vân.

Thật không may là chỉ có một chỗ trống duy nhất ở cuối hàng nên việc sắp xếp lại các container này chỉ có thể thực hiện theo cách là di chuyển container tới vị trí trống. Sau khi quá trình sắp xếp này hoàn thành chỗ trống phải nằm ở cuối hàng.

Hãy viết chương trình đưa ra số thao tác cần thực hiện ít nhất.

### Input

Dòng đầu tiên của file input chứa hai số nguyên N và M (1 ≤ M, N ≤ 400). Dòng thứ hai chứa N\*M số nguyên là nhãn của các container. Mỗi sản phẩm chiếm đúng N container.

### Output

Dòng đầu tiên của file output chứa số S, là số thao tác ít nhất cần thực hiện để nhận được vị trí các container như mong muốn. Sau đó là S dòng mô tả về các bước chuyển bằng hai số x y với ý nghĩa là chuyển container ở vị trí x tới vị trí y. Các vị trí được đánh số từ 1 tới N\*M + 1, vị trí trống ban đầu là N\*M+1. Di chuyển x tới y là hợp lệ nếu y là trống, sau khi di chuyển x sẽ trống. Nếu bài toán có nhiều nghiệm chỉ cần đưa ra một nghiệm.

### Ví dụ

|  |  |
| --- | --- |
| **Input** | **Output** |
| **5 6**  **4 1 3 1 6 5 2 3 2 3 5 6 2 1 4 5 6 4 1 3 2 4 5 5 1 2 3 4 6 6** | **8**  **9 31**  **18 9**  **10 18**  **4 10**  **31 4**  **30 31**  **24 30**  **31 24** |

## Bài 19: Cycle Detection

Giả sử chúng ta có một mạng máy tính mỗi máy tính kết nối với các máy tính khác trong mạng bằng cáp mạng. Mỗi đoạn cáp chỉ kết nối trực tiếp giữa hai máy tính và hai máy tính có thể kết nối với nhau chỉ bằng một đoạn cáp. Hãy xác định tất cả các đoạn cáp nằm trong một chu trình. Với mỗi đoạn cáp tham gia vào chu trình hãy xác định số lượng chu trình mà nó tham gia vào. Một chu trình là một dãy các dây cáp bắt đầu từ một máy tính A và cuối cùng lại kết thúc tại A. Tuy nhiên mỗi máy tính trừ A chỉ tham gia vào chu trình 1 lần duy nhất.

### Input

Dữ liệu của chương trình được cho trong một file text như sau: dòng đầu của file chứa một số nguyên N (N ≤ 20) là số máy tính trong mạng. Tiếp theo ma trận mô tả các kết nối trong mạng gồm N dòng, mỗi dòng gồm N số nguyên ngăn cách với nhau bằng một dấu cách.

### Output

Dữ liệu output của chương trình ghi vào một file text theo định dạng sau: dòng đầu là một số nguyên M tương ứng với số đoạn cáp mạng tham gia vào ít nhất một chu trình trong mạng, sau đó là dòng ghi M số nguyên sắp theo thứ tự tăng dần là số chu trình mà mỗi đoạn cáp tham gia vào. Nếu mạng không có chu trình thì kết quả là “No cycle”.

### Ví dụ

|  |  |
| --- | --- |
| **Input** | **Output** |
| **6**  **0 1 1 0 0 0**  **1 0 1 1 0 0**  **1 1 0 0 0 1**  **0 1 0 0 1 0**  **0 0 0 1 0 1**  **0 0 1 0 1 0** | **7**  **2 2 2 2 2 2 2** |
| **4**  **0 1 0 0**  **1 0 1 0**  **0 1 0 1**  **0 0 1 0** | **NO CYCLE** |

## Bài 20: FewestFactors

Cho một mảng các chữ số nhận các giá trị từ 0 tới 9 (không quá 5 chữ số). Hãy xây dựng một số nguyên từ các chữ số đó sao cho mỗi chữ số trong mảng xuất hiện một lần duy nhất và số ước số của số đó là ít nhất có thể được. Chú ý là chúng ta có thể sử dụng số 0 ở đầu, khi đó số 0 coi như bỏ.

### Input

Dữ liệu của chương trình được nhập từ 1 file text như sau: dòng đầu là số chữ số trong xâu (không vượt quá 5). Dòng thứ hai là số chữ số trong mảng.

### Output

Ghi kết quả xử lý vào một file text.

### Ví dụ

|  |  |
| --- | --- |
| **Input** | **Output** |
| **2**  **1 2** | **21** |
| **2**  **6 0** | **6** |
| **3**  **4 7 4** | **447** |
| **4**  **1 3 7 9** | **1973** |
| **5**  **7 5 4 3 6** | **36457** |

## Bài 21: PlayingCubes

Các khối hình hộp sử dụng trong một trò chơi dành cho trẻ em có các mặt được gán các nhãn là các chữ cái viết hoa. Mục đích của trò chơi là tạo thành các từ sử dụng các hình hộp như vậy. Chẳng hạn nếu chúng ta muốn xây dựng từ “DOG” chúng ta cần phải tìm 3 hình hộp một chứa chữ cái ‘D’, một chứa chữ cái ‘O’ và một chứa chữ cái ‘G’, sau đó xoay các hình này để các mặt chứa các chữ cái này đều là mặt trên của hình hộp.

Cho một danh sách các hình hộp có cùng thứ tự các mặt và danh sách các từ, hãy đưa ra số thứ tự của các từ có thể tạo thành theo nguyên tắc trên (các từ được đánh số từ 0).

### Input

Dữ liệu input của chương trình được cho trong một file text như sau: dòng đầu là 2 số N, M (2 ≤ N ≤ 8, 1 ≤ M ≤ 50) tương ứng là số hình hộp và số từ. Sau đó là N dòng ghi các mặt của N hình hộp và M dòng tiếp theo ghi các từ cần tìm (các từ đều có số ký tự nhỏ hơn 9).

### Output

Đưa ra danh sách thứ tự các từ tìm thấy.

### Ví dụ

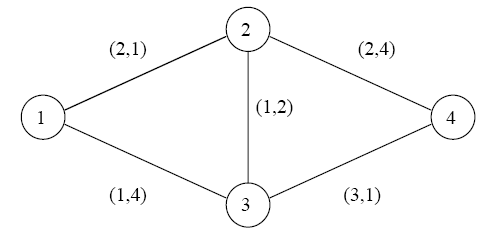
|  |  |
| --- | --- |
| **Input** | **Output** |
| **5 3**  **ABCDEF**  **DEFGHI**  **OPQRST**  **ZZZZZZ**  **YYYYYY**  **CAT**  **DOG**  **PIZZA** | **1** |
| **4 5**  **AAAAAA**  **AAAAAA**  **AAAAAA**  **AAAAAA**  **AA**  **AAA**  **AAAA**  **AAAAA**  **AAAAAA** | **0 1 2** |

## Bài 22: Bicritical routing

Hệ thống các đường cao tốc trả tiền ở Byteland đang phát triển rất nhanh chóng nên việc lựa chọn một hành trình (route) tốt nhất không phải là một công việc dễ dàng. Hệ thống đường cao tốc bao gồm các đường hai chiều kết nối giữa các thành phố. Mỗi con đường như vậy được mô hình hóa bằng thời gian di chuyển giữa chúng và chi phí phải trả khi đi trên đường.

Mỗi hành trình (đường đi) bao gồm một số đường cao tốc nối trực tiếp giữa các thành phố. Tổng thời gian cần thiết để đi hết 1 con đường là tổng số thời gian đi trên các đường cao tốc nối trực tiếp giữa các thành phố, tương tự chúng ta có định nghĩa về tổng chi phí cho mỗi con đường. Đường nào càng nhanh và có chi phí càng thấp thì càng được đánh giá tốt hơn. Một cách chặt chẽ hơn chúng ta có thể định nghĩa một đường đi là tốt hơn một đường đi khác nếu như có thể đi nhanh hơn và không phải trả nhiều hơn hay chúng ta sẽ trả ít hơn và đi không chậm hơn. Một đường đi tối ưu giữa hai thành phố là con đường tốt nhất nối giữa hai thành phố đó. Thật không may là không phải luôn tồn tại những con đường tối ưu như vậy – có thể có một số con đường nhưng chúng ta không thể so sánh chúng với nhau hoặc có thể không có con đường nào cả.

Hình vẽ sau là một minh họa hệ thống đường cao tốc giữa bốn thành phố, các trọng số trên các đường cao tốc này là chi phí và thời gian của các con đường.



Chúng ta có 4 con đường để đi từ 1 tới 4: 1-2-4 (chi phí 4, thời gian 5), 1-3-4 (chi phí 4, thời gian 5), 1-2-3-4 (chi phí 6, thời gian 4) và 1-3-2-4 (chi phí 4, thời gian 10).

Các đường đi 1-3-4 và 1-2-4 tốt hơn so với 1-3-2-4. Có các đường đi tối ưu sau: 1-2-4 và 1-3-4, có chi phí 4 thời gian 5 và đường đi 1-2-3-4 với chi phí 6 thời gian 4. Khi cần lựa chọn chúng ta sẽ căn cứ vào yêu cầu muốn đi theo đường đi có chi phí nhỏ hơn hay thời gian nhanh hơn.

Hãy viết chương trình tính xem có bao nhiêu đường đi tối ưu như vậy trong một hệ thống đường cao tốc. Các đường đi tối ưu có chi phí, thời gian giống nhau được tính là 1 đường đi.

### Input

Dữ liệu của chương trình được cho trong một file text như sau: dòng đầu của file là 4 số nguyên n, m, s, e (1 ≤ n ≤ 100, 0 ≤ m ≤ 300, 1≤ s, e ≤ n) trong đó n là số thành phố, m là số con đường, s và e là thành phố bắt đầu và kết thúc. Sau đó là m dòng mỗi dòng là một mô tả về một con đường cao tốc gồm 4 số nguyên là thành phố bắt đầu, kết thúc, chi phí và thời gian tương ứng với con đường cao tốc đó (chi phí và thời gian là các số tự nhiên nhỏ hơn 100).

### Output

Kết quả xử lý của chương trình ghi vào một file text.

### Ví dụ

|  |  |
| --- | --- |
| **Input** | **Output** |
| **4 5 1 4**  **2 1 2 1**  **3 4 3 1**  **2 3 1 2**  **3 1 1 4**  **2 4 2 4** | **2** |

## Bài 23: A Fast Journey

Hệ thống các thành phố được nối với nhau bằng các con đường một chiều. Các thành phố được đánh số từ 1 tới N (N ≤ 100). Vào những ngày lẻ các phương tiện di chuyển theo các hướng bình thường của các con đường và vào các ngày chẵn thì theo hướng ngược lại. Độ dài của các con đường được đo bằng một số nguyên là thời gian đi trên con đường đó và không phụ thuộc vào hướng di chuyển.

Hãy viết một chương trình tìm đường đi nhanh nhất từ một thành phố A tới thành phố B.

Ngày đầu tiên của hành trình là ngày lẻ, một ngày đi đường không quá 12 tiếng và đêm phải nghỉ tại một thành phố, hành trình sẽ được tiếp tục vào ngày hôm sau.

### Input

Dữ liệu input của chương trình được cho trong một file text như sau: dòng đầu tiên là hai số nguyên tương ứng với số thứ tự của A và B, dòng thứ hai là hai số nguyên tương ứng với số thành phố và số con đường một chiều (k) nối giữa các thành phố đó (số thành phố nhỏ hơn 1000). Tiếp theo là k dòng môt tả về các con đường, mỗi dòng gồm 3 số là thành phố bắt đầu, kết thúc và thời gian để đi hết con đường.

### Output

Kết quả xử lý của chương trình ghi vào một file text theo các dòng là mô tả về các con đường một chiều trên hành trình từ A tới B, mỗi mô tả gồm 4 số nguyên tương ứng là thành phố bắt đầu, thành phố đến, ngày đi và thời gian đi hết con đường đó.

### Ví dụ

|  |  |
| --- | --- |
| **Input** | **Output** |
| **1 3**  **6 7**  **1 2 9**  **1 6 2**  **1 5 10**  **5 4 1**  **4 6 2**  **4 3 4**  **2 3 5** | **1 5 1 10**  **5 4 1 1**  **4 3 3 4** |

1

2

3

4

5

6

Hình vẽ minh họa:

## Bài 24: Speed Limits

Trong công việc hàng ngày chúng ta có thể không quan tâm tới các con đường ngắn nhất mà thường là những con đường có thời gian ngắn nhất. Khi lái xe thì vấn đề giới hạn tốc độ trên các con đường là rất quan trọng đặc biệt là khi bộ giao thông vận tải trang bị cho các chú cảnh sát giao thông các súng bắn tốc độ. Tuy nhiên đôi khi vẫn có các biển báo tốc độ bị mờ mà chúng ta lại không thể nhớ hết tất cả các giới hạn tốc độ tại các con đường. Chúng ta chỉ có thể giả sử rằng khi rẽ vào một đường khác tốc độ hiện tại sẽ không bị giảm. Hãy viết chương trình tìm đường đi nhanh nhất dựa việc tận dụng các con đường mất biển báo tốc độ.

Chúng ta giả sử tất cả các con đường đều là một chiều, hệ thống đường gồm các ngã rẽ và các đường nối, một đường chỉ nối giữa hai ngã rẽ, mỗi con đường đều có nhiều nhất là một biển báo tốc độ ở đầu đường

### Input

Dòng đầu tiên của file input chứa 3 số nguyên N, M và D (2 ≤ N ≤ 150) là số các ngã rẽ đánh số từ 0 tới N-1, M là số con đường và D là ngã rẽ cần đi đến. Tiếp theo là M dòng, mỗi dòng mô tả về 1 con đường gồm 4 số A, B, V và L trong đó A là điểm xuất phát B là đích, giới hạn tốc độ là V và độ dài L. Nếu như V bằng 0 nghĩa là biển báo tốc độ bị mờ. Điểm xuất phát là ngã rẽ 0 với tốc độ hiện tại là 70.

### Output

Kết quả xử lý của chương trình là số thứ tự của các ngã rẽ trên đường đi nhanh nhất tìm được. Giả sử chúng ta không bao giờ có 2 đường nhanh nhất có cùng thời gian.

### Ví dụ

|  |  |
| --- | --- |
| **Input** | **Output** |
| **6 15 1**  **0 1 25 68**  **0 2 30 50**  **0 5 0 101**  **1 2 70 77**  **1 3 35 42**  **2 0 0 22**  **2 1 40 86**  **2 3 0 23**  **2 4 45 40**  **3 1 64 14**  **3 5 0 23**  **4 1 95 8**  **5 1 0 84**  **5 2 90 64**  **5 3 36 40** | **0 5 2 3 1** |

## Bài 25: MoneyExchange

Bạn có một số tiền loại type1 và muốn đổi sang tiền loại type2. Tuy nhiên tại cửa khẩu đôi khi không thể đổi trực tiếp từ loại tiền này sang loại tiền khác nên đôi khi chúng ta cần phải tiến hành đổi một số lần qua các loại tiền trung gian khác. Tất nhiên các nhân viên đổi tiền đủ thông minh để không cho phép chúng ta đổi quay vòng nhiều lần cùng một loại tiền để thu được lợi nhuận.

Hãy viết chương trình xác định xem chúng ta có thể nhận được tối đa bao nhiêu tiền type2 nếu có 1 đơn vị tiền type1. Nếu không thể đổi được thì kết quả la -1.

### Input

Dữ liệu của chương trình được cho trong một file text như sau: dòng đầu là số tỉ giá đổi tiền N (0 ≤ N ≤ 50), sau đó là N dòng mỗi dòng ghi tỉ giá đổi tiền theo dạng t1 m1 t2 m2 với ý nghĩa là m1 đồng tiền t1 đổi được m2 đồng tiền t2. Tên các loại tiền là các xâu viết hoa không vượt quá 50 ký tự. Cuối cùng là hai dòng ghi tên type1 và type2.

### Output

Ghi kết quả xử lý của chương trình vào một file text.

### Ví dụ

|  |  |
| --- | --- |
| **Input** | **Output** |
| **1**  **EUR 0008 USD 0010**  **USD**  **EUR** | **1.25** |
| **3**  **EUR 8 USD 10**  **EUR 1 GOODBUY 2**  **GOODBUY 1 USD 2**  **EUR**  **USD** | **4.0** |
| **5**  **A 1 B 2**  **A 1 C 3**  **B 2 D 3**  **C 9 D 10**  **D 7 E 2**  **A**  **E** | **0.952** |

## Bài 26: 106 miles from Chicago

Hai điệp vụ đang làm nhiệm vụ cho chính phủ và họ cần chuyển một tài liệu về Chicago. Các cơ quan tình báo nước ngoài cũng đang tìm cách lấy tài liệu này và họ đã giăng lưới trên các con đường dấn tới Chicago để đón lõng hai điệp vụ. Hãy viết chương trình tìm con đường an toàn nhất cho hai điệp vụ.

### Input

Dữ liệu input của chương trình được cho trong một file text và có thể có nhiều test. Mỗi test được cho như sau: dòng đầu là hai số nguyên n, m (2 ≤ n ≤ 100, 1 ≤ m ≤ (n-1)\*n/2) trong đó m là số ngã rẽ và m là số con đường nối giữa các ngã rẽ. Tiếp theo mà m dòng mô tả về các con đường, mỗi dòng gồm 3 số nguyên a, b, p trong đó a là ngã rẽ bắt đầu, b là kết thúc và p là sác xuất không bị bắt của hai điệp vụ nếu đi trên con đường đó. Các ngã rẽ được đánh số từ 1.

### Output

Với mỗi test hãy tìm con đường an toàn nhất đi từ ngã rẽ 1 tới ngã rẽ N. Ghi kết quả là sác xuất an toàn của con đường vào 1 file text.

### Ví dụ

|  |  |
| --- | --- |
| **Input** | **Output** |
| **5 7**  **5 2 100**  **3 5 80**  **2 3 70**  **2 1 50**  **3 4 90**  **4 1 85**  **3 1 70**  **0** | **61.200** |

## Bài 27: LongBlob

Các bức ảnh chụp từ vệ tinh bao gồm các hình vuông kích thước 1x1 tạo thành môt bảng hình chữ nhật, mỗi hình vuông có giá trị bằng 0 hoặc 1. Các hình vuông chứa số 0 có chung cạnh tạo thành một vùng gọi là một blob. Độ dài của một blob là khoảng cách lớn nhất giữa hai tâm của bất cứ cặp các ô vuông chứa số 0 nào trong blob đó. Chẳng hạn bức ảnh sau:

**0010**

**1001**

**0011**

**0111**

Có 2 blob và độ dài của blob lớn hơn là sqrt(1 + 9) là khoảng cách giữa ô vuông đầu tiên của hàng cuối cùng với số 0 ở hàng đầu tiên cột thứ hai.

Vấn đề là các bức ảnh có thể có nhiễu nên một số hình có giá trị 1 thực sự phải mang giá trị 0. Chúng ta không quan tâm tới việc phục hồi chính xác mà quan tâm tới việc nếu như đổi 4 ô mang giá trị 1 thành các ô mang giá trị 0 thì vùng lớn nhất sẽ có độ dài là bao nhiêu.

### Input

Dữ liệu input của chương trình được cho như sau: dòng đầu chứa 2 số nguyên N, M là kích thước của bước ảnh (1 ≤ N, M ≤ 25). Sau đó là N dòng, mỗi dòng chứa M số hoặc 0 hoặc 1.

### Output

Ghi kết quả tìm được vào 1 file text.

### Ví dụ

|  |  |
| --- | --- |
| **Input** | **Output** |
| **4 4**  **0010**  **1001**  **0011**  **0111** | **4.2426** |
| **1 9**  **101010101** | **7.0** |

## Bài 28: Fence Loops

Các hàng rào bao quanh các khu chăn thả của Mr Brown được tạo thành từ các đoạn rào thẳng ngắn hơn có độ dài từ 1 tới 200 feet và chỉ nối với nhau tại các điểm đầu và cuối của chúng (có thể có nhiều đoạn rào nối với nhau tại một điểm đầu cuối). Kết quả là có một mạng các hàng rào tạo thành từ các đoạn rào bao quanh các khu chăn thả. Mr Brown muốn điểu chỉnh lại các đoạn chắn này cho phù hợp và ông ta muốn biết phần nào của khu chăn thả có chu vi nhỏ nhất (Mỗi khu chăn thả bị bao quanh bởi 1 số các đoạn chắn).

Các đoạn chắn được đánh số bắt đầu từ 1 tới N (N là tổng số các đoạn chắn), với mỗi đoạn chắn Mr Brown biết các thông tin sau:

* Độ dài của đoạn chắn
* Các đoạn chắn khác nối với hai đầu của nó
* Không có đoạn chắn nào tự nối với chính nó

Cho một danh sách các đoạn chắn bao quanh các khu chăn thả hãy viết chương trình giúp Mr Brown tính xem khu chăn thả nào có chu vi nhỏ nhất.

Chẳng hạn đối với khu chăn thả như hình vẽ sau;

**1**

**+---------------+**

**|\ /|**

**2| \7 / |**

**| \ / |**

**+---+ / |6**

**| 8 \ /10 |**

**3| \9 / |**

**| \ / |**

**+-------+-------+**

**4 5**

Thì khu chăn thả có chu vi nhỏ nhất chính là khu chăn thả tạo thành từ 3 đoạn chắn có số thứ tự là 2, 7 và 8.

### Input

Dữ liệu của chương trình được cho trong một file text có định dạng như sau:

Dòng 1: là một số nguyên N tương ứng với số test trong file (1 ≤ N ≤ 100)

Tiếp đến là N bộ ba dòng mô tả về các đoạn chắn như sau: dòng đầu là 4 số nguyên s (1 ≤ s ≤ N) là số hiệu của đoạn chắn, Ls (1 ≤ Ls ≤ 255) là độ dài đoạn chắn, N1s (1 ≤ N1s ≤ 8) là số phần tử trên dòng thứ hai, N2s (1 ≤ N2s ≤ 8) là số phần tử trên dòng thứ ba. Dòng thứ hai chứa N1s số nguyên là các đoạn chắn có nối với đoạn chắn hiện tại ở một đầu. Dòng thứ 3 chứa N2s số nguyên là các đoạn chắn có nối với đoạn chắn hiện tại ở đầu kia.

### Output

Với mỗi test hãy in ra chu vi của các đoạn chắn bao quanh vùng chăn thả nhỏ nhất.

### Ví dụ

|  |  |
| --- | --- |
| **Input** | **Output** |
| **10**  **1 16 2 2**  **2 7**  **10 6**  **2 3 2 2**  **1 7**  **8 3**  **3 3 2 1**  **8 2**  **4**  **4 8 1 3**  **3**  **9 10 5**  **5 8 3 1**  **9 10 4**  **6**  **6 6 1 2**  **5**  **1 10**  **7 5 2 2**  **1 2**  **8 9**  **8 4 2 2**  **2 3**  **7 9**  **9 5 2 3**  **7 8**  **4 5 10**  **10 10 2 3**  **1 6**  **4 9 5** | **12** |

## Bài 29: Nagibni

Hệ thống đường sắt của mỗi quốc gia thường bao gồm một số tuyến đường sắt nhất định. Mỗi tuyến đường sắt thường nối giữa hai thành phố khác nhau và giữa hai thành phố chỉ có một tuyến đường sắt trực tiếp nối giữa chúng. Gần đây các nhà quản lý hệ thống đường sắt đã quyết định thay thế hệ thống đường sắt cũ kỹ của họ bằng một hệ thống đường ray mới, nhanh và hiện đại hơn được nhập khẩu từ Nhật Bản.

Một hành trình sẽ đi qua ít nhất 2 thành phố, các thành phố liền kề nhau trên mỗi hành trình sẽ được nối với nhau bằng một tuyến đường sắt. Độ dài của hành trình tính bằng tổng độ dài của các tuyến đường trên hành trình đó. Vì tốc độ của các đoàn tàu giờ đây đã rất nhanh nên không cho phép hai hành trình cùng đi qua một thành phố.

Hãy viết một chương trình xác định các hành trình chạy tàu của hệ thống đường sắt mới với các điều kiện sau: chỉ có một hành trình đi qua mỗi thành phố và tổng độ dài của tất cả các hành trình là lớn nhất có thể được.

### Input

Dòng đầu của file input chứa một số nguyên N (1 ≤ N ≤ 2000) là số lượng các thành phố. Các thành phố được đánh số từ 1 tới N.

Tiếp theo là N – 1 dòng chứa mô tả về các tuyến đường nối giữa các thành phố. Mỗi dòng chứa 3 số nguyên A, B, C với ý nghĩa là tuyến đường nối thành phố A với thành phố B có độ dài là C (1 ≤ C ≤ 1000000).

Chú ý: Dữ liệu input của bài toán luôn đảm bảo tồn tại 1 nghiệm.

### Output

Ghi vào file input tổng độ dài lớn nhất tìm được.

### Ví dụ

|  |  |
| --- | --- |
| **Input** | **Output** |
| **5**  **1 2 10**  **1 3 10**  **1 4 10**  **4 5 1** | **21** |
| **6**  **1 2 1**  **1 3 3**  **1 4 7**  **2 5 4**  **2 6 6** | **20** |
| **9**  **1 2 1**  **2 4 2**  **3 4 1**  **4 5 5**  **6 5 1**  **5 7 2**  **7 8 1**  **8 9 2** | **11** |

## Bài 30: Water Jugs

Vợ của Mr John, Mrs Martha cần một lượng nước để làm nước chanh. Tuy nhiên Msr Martha cần một lượng nước chính xác vì nếu không có thể làm cho nước chanh không ngon. Msr Martha có một vài bình đựng nước và có thể đong được lượng nước cần thiết bằng các thao tác sau đây:

1. Đổ đầy nước vào một bình (nước được lấy từ 1 bể chứa)
2. Đổ nước từ bình 1 sang bình 2 cho tới khi bình 1 đầy hoặc bình 2 hết.
3. Đổ hết nước ra khỏi 1 bình nào đó (đổ ra cống thoát nước).

Hãy viết chương trình giúp Msr Martha tính xem cần ít nhất bao nhiêu thao tác để có thể có được lượng nước mong muốn trong một bình nào đó. Nếu có nhiều giải pháp hãy đưa ra giải pháp sử dụng ít lượng nước nhất (lượng nước lấy từ bể).

Chẳng hạn nếu Mrs Martha có 3 bình lần lượt có dung tích là 6, 7, và 10 và cần 8 lit nước thì có thể tiến hành như sau:

1. Đổ đầy bình 7 lit
2. Đổ hết bình 7 lit dang bình 10 lit
3. Đổ đầy bình 7 lit
4. Đổ nước từ bình 7 lit sang bình 6 lit, bình 7 lit sẽ còn lại 1 lit
5. Đổ 1 lit từ bình 7 lit sang bình 10 lit.

### Input

Dòng đầu của file input chứa hai số nguyên N và L trong đó N là số lượng các bình và L là số lượng nước cần lấy. Dòng thứ hai chứa đúng N số nguyên là dung tích của N bình đựng nước mà Mrs Martha có. (1 ≤ N ≤ 6, 1 ≤ K ≤ 11).

### Output

Dòng đầu tiên của file output chứa hai số nguyên O và W trong đó O là số thao tác và W là số nước cần dùng. Tiếp theo là O dòng, mỗi dòng chứa hai số nguyên S và D mô tả các thao tác thực hiện, trong đó S và D là số hiệu của các bình nước trong thao tác tương ứng (các bình nước được đánh số từ 0 tới N – 1). Nếu như đó là một thao tác đổ đầy bình thì S sẽ bằng -1 còn nếu đó là thao tác đổ hết nước khỏi bình (đổ ra cống thoát) thì D sẽ bằng -1.

### Ví dụ

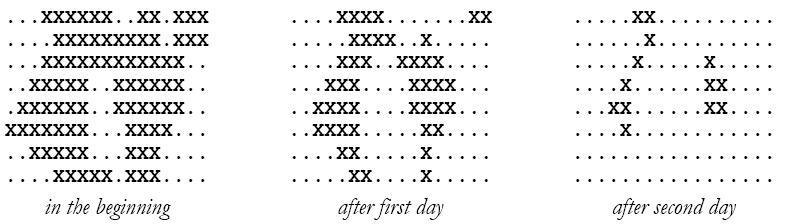
|  |  |
| --- | --- |
| **Input** | **Output** |
| **3 8**  **6 7 10** | **5 14**  **-1 1**  **1 2**  **-1 1**  **1 0**  **1 2** |

## Bài 31: Labudovi

Hai con thiên nga cùng sống trong một hồ nước nhưng bị ngăn cách với nhau bởi các tảng băng trên mặt hồ. Hồ nước có hình dạng một hình chữ nhật được chia thành R hàng và C cột, với mỗi ô là nước hoặc băng.

Mùa hè đang đến và băng trên hồ dần tan. Sau mỗi ngày các phần băng tiếp xúc với nước sẽ bị bốc hơi (các ô chứa băng tiếp xúc với các ô chứa nước sẽ bị tan chảy). Hai ô được gọi là tiếp xúc với nhau nếu như chúng có chung cạnh.

Chẳng hạn băng trên hồ (ký hiệu là các chữ ‘x’) sẽ tan như các hình vẽ sau:



Các chú thiên nga chỉ có thể di chuyển trên mặt hồ qua các ô chứa nước chung cạnh.

Hãy viết chương trình tính xem sau bao nhiêu ngày thì hai chú thiên nga có thể gặp nhau.

### Input

Dòng đầu tiên của file input chứa hai số nguyên R và C, 1≤ R, C ≤ 1500. Tiếp theo là R dòng, mỗi dòng chứa C ký tự hoặc ‘**.**’ có nghĩa là nước, ‘X’ có nghĩa là băng và ‘L’ có nghĩa là vị trí của thiên nga.

### Output

Hãy đưa ra số ngày mà sau đó các chú thiên nga có thể gặp nhau.

### Ví dụ

|  |  |
| --- | --- |
| **Input** | **Output** |
| **10 2**  **.L**  **..**  **XX**  **XX**  **XX**  **XX**  **XX**  **XX**  **..**  **.L** | **3** |
| **4 11**  **..XXX...X..**  **.X.XXX...L.**  **....XXX..X.**  **X.L..XXX...** | **2** |
| **8 17**  **...XXXXXX..XX.XXX**  **....XXXXXXXXX.XXX**  **...XXXXXXXXXXXX..**  **..XXXXX.LXXXXXX..**  **.XXXXXX..XXXXXX..**  **XXXXXXX...XXXX...**  **..XXXXX...XXX....**  **....XXXXX.XXXL...** | **2** |

## Bài 32: Teleport

Nhà phù thủy đại tài Byter đã tạo ra hai hòn đảo trên biển Baltic: Bornholm và Gotland. Trên mỗi hòn đảo này có một số máy truyền tín hiệu đặc biệt (teleport). Mỗi máy truyền tín hiệu này có thể hoạt động ở 1 trong 2 chế độ sau:

* Nhận tín hiệu từ một máy khác truyền tới nó
* Gửi tín hiệu tới một máy khác (ở khác đảo với nó) miễn là máy đó đang làm việc ở chế độ nhận tín hiệu.

Một lần Byter đã đưa ra yêu cầu sau cho các môn đệ của mình: họ phải đặt chế độ làm việc cho các máy truyền tín hiệu sao cho không có máy nào không được sử dụng, mỗi máy làm việc ở chế độ nhận sẽ có ít nhất 1 máy truyền tín hiệu tới cho nó và ngược lại với mỗi máy truyền tín hiệu, máy nhận tín hiệu từ nó cũng phải làm việc ở chế độ nhận.

Hãy viết chương trình xác định chế độ hoạt động của các máy truyền tín hiệu ở trên hai hòn đảo theo yêu cầu của Mr Byter. Nếu bài toán có nhiều nghiệm chỉ cần đưa ra một nghiệm.

### Input

Dòng đầu của file input chứa hai số nguyên m và n (1 ≤ m, n ≤ 50000) trong đó m là số máy truyền tín hiệu trên đảo Bornholm và n là số máy trên đảo Gotland. Các máy trên các đảo được đánh số từ 1 tới m và từ 1 tới n tương ứng. Dòng thứ hai chứa m số nguyên không lớn hơn n trong đó số thứ k tương ứng là số hiệu của các máy nhận tín hiệu trên đảo Gotland của máy thứ k trên đảo Bornholm. Dòng thứ 3 chứa n số nguyên không lớn hơn m tương ứng là số thứ tự của các máy nhận tín hiệu trên đảo Bornholm của các máy trên đảo Gotland.

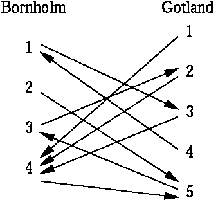
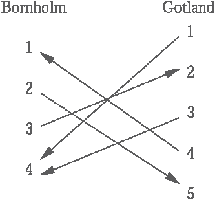
### Output

Kết quả xử lý của chương trình cần đưa ra là chế độ làm việc của các máy truyền tín hiệu trên 2 đảo bằng 2 xâu bít ghi trên 2 dòng của file output. Trong đó máy thứ k trên mỗi đảo sẽ được cho bằng số 0 nếu hoạt động ở chế độ nhận tín hiệu còn 1 nghĩa là gửi tín hiệu.

### Ví dụ

|  |  |
| --- | --- |
| **Input** | **Output** |
| **4 5**  **3 5 2 5**  **4 4 4 1 3** | **0110**  **10110** |

### Hình minh họa

**Input Output**

## Bài 33: Hamming Numbers

Các số Hamming lần đầu tiên được đưa ra trong một bài tập của Richard W. Hamming, người đã phát minh ra các mã Hamming. Theo định nghĩa một số Hamming là một số nguyên dương có thể phân tích thành tích của một số các thừa số được chọn ngẫu nhiên nào đó. Chẳng hạn nếu tập các thừa số là {2, 3, 5} thì 90 = 2\*3\*3\*5 là một số Hamming nhưng 70 = 2\*5\*7 lại không phải là một số Hamming vì nó còn chia hết cho 7 (không nằm trong tập các thừa số). Số 1 luôn được xem là một số Hamming cho dù các thừa số được chọn như thế nào đi chăng nữa.

Hãy viết chương trình xác định số Hamming thứ n với một tập các thừa số cho trước.

### Input

Dòng đầu của file input là một tập các thừa số ngăn cách với nhau bằng 1 dấu cách, số các thừa số không vượt quá 50, một thừa số có giá trị lớn hơn 1 và nhỏ hơn 300. Dòng thứ hai là số N (1 ≤ N ≤ 100000).

### Output

Đưa ra kết quả là số Hamming thứ N.

### Ví dụ

|  |  |
| --- | --- |
| **Input** | **Output** |
| **2 3 5**  **15** | **24** |
| **2 2 2 4 4 4 8 8 8**  **11** | **1024** |
| **7 9 14 6**  **52** | **4802** |

## Bài 34: The Mysterious X network

Một trong những lý do mà trường Bách khoa Ecole có ảnh hưởng lớn trong xã hội Pháp là sự nổi tiếng của hệ thống mạng các camarades – các sinh viên cũ của trường. Khi một camarades muốn giúp đỡ (về mặt tiền bạc, công việc) họ có thể nhận được sự hỗ trợ từ mạng này. Trên thực tế khi muốn gặp một camarades nào đó họ có thể tiến hành qua các camarades trung gian để nhận được sự giúp đỡ.

Hãy viết chương trình xác định số các camarades trung gian ít nhất khi một người nào đó cần giúp đỡ.

### Input

Dữ liệu chương trình được cho như sau: dòng đầu của file input là một số nguyên N (1 ≤ N ≤ 105). Các camarades được đánh số từ 0 tới N-1. Tiếp sau đó là N dòng mỗi dòng bắt đầu bằng số thứ tự của camarades tương ứng (C) sau đó là số các camarades mà người đó quen biết (NC), tiếp đến là NC số hiệu của các camarades quen C. Dòng cuối cùng là 2 số C1 và C2 trong đó C1 là người gặp khó khăn và C2 là người sẵn sàng giúp anh ta.

### Output

In ra 3 số C1, C2 và số người trung gian ít nhất để C1 có thể gặp C2.

### Ví dụ

|  |  |
| --- | --- |
| **Input** | **Output** |
| **4**  **0 3 1 2 3**  **1 1 0**  **2 2 0 3**  **3 2 0 2**  **1 2** | **1 2 1** |

## Bài 35: FILL

Có 3 bình đựng nước có dung tích là a, b và c lít (là các số nguyên không lớn hơn 200). Bình đầu tiên và bình thứ 2 trống trong khi bình thứ 3 đầy nước. Có thể đổ nước từ bình này sang bình khác cho tới khi bình thứ nhất cạn hoặc bình thứ 2 đầy.

Hãy viết chương trình xác định số lượng nước cần rót ít nhất để sao cho có ít nhất 1 bình nào đó chứa đúng d lít nước (d là một số nguyên dương nhỏ hơn 200). Nếu như không thể đong được chính xác d lít nước, chương trình cần đưa ra dung tích nhỏ hơn và gần với d nhất có thể được (gọi là d’) và số lit nước rót nhỏ nhất để đạt được kết quả đó.

### Input

File input được cho như sau: dòng đầu tiên là một số nguyên T tương ứng với số test trong file. Tiếp theo là T dòng mỗi dòng mô tả một test gồm 4 số nguyên a, b, c, d.

### Output

Kết quả mỗi test ghi trên 1 dòng gồm 2 số: số đầu là d hoặc d’ số thứ hai là số lít nước cần rót.

### Ví dụ

|  |  |
| --- | --- |
| **Input** | **Output** |
| **4**  **2 3 4 2**  **96 97 199 62**  **2 5 5 5**  **8 7 6 7** | **2 2**  **9859 62**  **0 5**  **0 6** |

## Bài 36: The Domino Train

Một dãy domino được xếp từ các quân domino mà các quân domino liên tiếp nhau cần tuân thủ điều kiện là các số ở 2 đầu tiếp xúc nhau của chúng phải bằng nhau. Dãy xếp được coi là tốt nhất nếu như tổng các số trên các quân domino là lớn nhất. Cho T quân domino hãy viết chương trình xác định tổng lớn nhất có thể được khi xếp một dãy domino từ các quân domino cho trước với điều kiện số các quân domino tham gia vào dãy không vượt quá N.

### Input

Dữ liệu của chương trình được cho trong một file text: dòng đầu là 2 số T, N (1 ≤ T, N ≤ 150) sau đó là T dòng mô tả về các quân domino, mỗi quân được mô tả trên 1 dòng gồm 2 số nguyên tương ứng ở hai đầu của quân domino.

### Output

Đưa ra tổng lớn nhất có thể được.

### Ví dụ

|  |  |
| --- | --- |
| **Input** | **Output** |
| **4 6 5 1 5 4 3 2 6 4** | **25**  **(1|5 5|4 4|6)** |

## Bài 37: Tennis Club (not available)

Một câu lạc bộ Tennis muốn tổ chức một tuần lễ quảng cáo thu hút những người chơi mới gia nhập vào câu lạc bộ bằng cách mời một số ngôi sao quần vợt về đánh biểu diễn. Mỗi ngôi sao sẽ cho biết họ số trận mà họ sẵn sàng tham gia. Các nhà tổ chức muốn cho các trận đấu hấp dẫn hơn nên không muốn có trường hợp 2 ngôi sao gặp nhau nhiều hơn 1 lần.

Hãy viết chương trình sắp xếp các ngôi sao tham gia các trận đấu sao cho mỗi ngôi sao đều tham dự đúng số trận mà họ thích đồng thời không có cặp nào gặp nhau nhiều hơn 1 lần.

### Input

Dòng đầu tiên của file input là số nguyên N (2 ≤ N ≤ 1000) và sau đó là N dòng, mỗi dòng là số trận tương ứng mà một ngôi sao muốn chơi.

### Output

Dòng đầu tiên của file output sẽ là thông báo có nghiệm hoặc vô nghiệm, nếu có nghiệm thì sau dó là N dòng, mỗi dòng là danh sách các đấu thủ mà một ngôi sao sẽ đấu.

### Ví dụ

|  |  |
| --- | --- |
| **Input** | **Output** |
| **3**  **1**  **2**  **1** | **Co nghiem**  **2**  **1 3**  **2** |
| **3**  **2**  **2**  **1** | **Vo nghiem** |

## Bài 38: Excursion (not available)

Một nhóm sinh viên quyết định chung tiền để đi du lịch tới một số thành phố. Mỗi sinh viên sẽ đưa ra 2 mong muốn, mong muốn thứ nhất là thành phố mà họ muốn đến, mong muốn thứ hai là thành phố mà họ không muốn đến (hai thành phố này có thể trùng nhau).

Hãy viết chương trình đưa ra danh sách các thành phố sẽ đến thăm của cả nhóm (có thể là rỗng) sao cho ít nhất một mong muốn của mỗi sinh viên đều được thỏa mãn. Nếu bài toán có nhiều nghiệm có thể đưa ra bất cứ nghiệm nào.

### Input

Dòng đầu của file input chứa hai số nguyên n và m (1 ≤ n ≤ 20000, 1 ≤ m ≤ 8000) trong đó n là số sinh viên, m là số thành phố. Các sinh viên được đánh số từ 1 tới n và các thành phố được đánh số từ 1 tới m. Tiếp theo là n dòng, mỗi dòng chứa hai số nguyên khác 0 ngăn cách với nhau bằng một dấu cách. Dòng thứ i + 1 chứa wi và wi’ thể hiện mong muốn của sinh viên thứ i trong đó –m ≤ wi, wi’ ≤ m, wi, wi’ ≠ 0. Một số nguyên dương có nghĩa là sinh viên đó muốn đến thăm thành phố tương ứng còn một số âm có nghĩa là sinh viên đó không muốn đến thăm thành phố bằng trị tuyệt đối của số đó.

### Output

Kết quả chương trình ghi vào một file text gồm có: dòng đầu là số thành phố của hành trình M, dòng thứ 2 là M số tương ứng với các thành phố trong hành trình. Nếu bài toán không có nghiệm thì kết quả là “Vo nghiem”.

### Ví dụ

|  |  |
| --- | --- |
| **Input** | **Output** |
| **3 4**  **1 -2**  **2 4**  **3 1** | **4**  **1 2 3 4** |

## Bài 39:

### Input

### Output

### Ví dụ

|  |  |
| --- | --- |
| **Input** | **Output** |
| **4**  **2 4 1**  **3 1 2 4**  **1 2**  **4** | **Co the truy cap**  **1 2 4**  **Khong the truy cap**  **3** |

# III. Dynamic Programming

## Bài 1: BikeLock

Các loại cặp số thường sử dụng một loại khóa số gồm một số vòng với các số từ 0 tới 9. Để mở khóa chúng ta cần xoay các vòng số này khớp với mã bí mật được thiết lập trước. Mỗi thao tác xoay có thể xoay vòng hiện tại theo chiều kim đồng hồ hoặc ngược chiều kim đồng hồ tới một số bất kỳ của vòng số cách nó nhiều nhất 3 đơn vị, một thao tác xoay cũng có thể áp dụng cho tối đa 3 vòng số liên tiếp nhau đồng thời xoay theo cùng chiều. Ví dụ nếu khóa bí mật là 363 và hiện tại các vòng là 555 chúng ta có thể tiến hành mở khóa bằng cách xoay vòng 1 về 3 bằng 1 thao tác xoay cùng chiều kim đồng hồ, sau đó xoay vòng 2 ngược chiều kim đồng hồ về 6, cuối cùng là xoay vòng 3 cùng chiều kim đồng hồ về số 3. Tuy nhiên cũng có thể dùng một cách khác là xoay cả 3 vòng về 333 sau đó xoay vòng 2 về 6.

Hãy viết chương trình xác định số thao tác ít nhất cần thiết để mở một khóa như vậy.

### Input

Dữ liệu của chương trình cho trong 1 file text gồm 2 dòng ghi các số của trạng thái hiện tại của khóa và mã số bí mật.

### Output

Đưa ra số thao tác ít nhất cần thiết để mở khóa.

### Ví dụ

|  |  |
| --- | --- |
| **Input** | **Output** |
| **555**  **464** | **2** |
| **1234**  **3456** | **2** |
| **123456789**  **567412490** | **5** |

## Bài 2: BadSubstrings

Các xâu được thành lập từ các ký tự A, B và C. Một xâu được gọi là xâu con của xâu khác nếu nó xuất hiện (không nhất thiết có độ dài nhỏ hơn) hoàn toàn trong một xâu khác. Chẳng hạn xâu “AB” là xâu con của xâu “ABC” nhưng không phải là xâu con của xâu “ACB”. Hãy tìm số các xâu có độ dài N nhưng không có xâu con nào là xâu b1 và xâu b2 trong đó b1 và b2 là các xâu cho trước.

### Input

Dữ liệu của chương trình được cho trong một file text như sau: dòng đầu là số N (1 ≤ N ≤ 39), sau đó là hai dòng ghi hai xâu b1 và b2.

### Output

Ghi số lượng các xâu thỏa mãn yêu cầu của bài toán.

### Ví dụ

|  |  |
| --- | --- |
| **Input** | **Output** |
| **3**  **AB**  **BA** | **17** |
| **2**  **A**  **BB** | **3** |
| **5**  **ABC**  **ABB** | **189** |

## Bài 3: IntegerSequence

Với một dãy các số nguyên từ đó chúng ta cần tạo ra một dãy con có độ dài lớn nhất thỏa mãn điều kiện sau: nó có thể cắt thành hai phần có chung một phần tử là phần tử cuối của phần thứ nhất và phần tử đầu của phần thứ hai sao cho phần thứ nhất được sắp tăng dần và phần thứ hai được sắp giảm dần. Chẳng hạn dãy {1, 4, 6, 5, 2, 1} có thể cắt thành hai dãy {1, 4, 6} và {6, 5, 2, 1} thỏa mãn yêu cầu trên.

Hãy viết chương trình xác định xem số lượng ít nhất các phần tử của một dãy cho trước cần bỏ đi là bao nhiêu để nhận được kết quả là một dãy thỏa mãn yêu cầu trên.

### Input

Dữ liệu của chương trình được cho trong 1 file text như sau: dòng đầu là một số nguyên N ( 1 ≤ N ≤ 50). Sau đó là N số trên dòng thứ hai, mỗi số ngăn cách với nhau bằng 1 dấu cách.

### Output

Kết quả chương trình ghi vào 1 file text.

### Ví dụ

|  |  |
| --- | --- |
| **Input** | **Output** |
| **6**  **1 4 6 5 2 1** | **0** |
| **9**  **1 2 1 2 3 2 1 2 1** | **4** |
| **5**  **2 2 2 2 2** | **4** |

## Bài 4: Beef McNuggets

Các loại rau quả trong tương lai sẽ được đóng gói vào các hộp chia theo đơn vị calo mà lượng rau cung cấp (là các số nguyên). Tuy nhiên cách đóng gói này cũng có sự bất tiện của nó. Chẳng hạn nếu chúng ta chỉ bán các gói có lượng calo cung cấp là 3, 6, 10 chẳng hạn chúng ta sẽ không thể bán cho những người có nhu cầu mua 1, 2, 4, 5, 7, 8, 11, 14, 17 calo.

Hãy viết chương trình tính xem với các gói rau xanh có lượng calo cố định như vậy thì lượng calo lớn nhất người dùng yêu cầu mà chúng ta không thể đáp ứng được là bao nhiêu.

### Input

Dữ liệu của chương trình được cho trong 1 file text như sau: dòng đầu là một số N (1 ≤ N ≤ 10) là số các loại hộp rau quả. Sau đó là N dòng mỗi dòng ghi một số nguyên dương là lượng calo của loại hộp tương ứng.

### Output

Ghi kết quả xử lý của chương trình vào 1 file text.

### Ví dụ

|  |  |
| --- | --- |
| **Input** | **Output** |
| **3**  **3**  **6**  **10** | **17** |

## Bài 5: Bugs Integrated, Inc

Vtel là một công ty sản xuất chip hàng đầu của Uland, họ sử dụng các mainboard là các bảng hình chữ nhật được chia nhỏ thành các ô vuông kích thước 1x1. Trên các ô vuông này có thể đã có các chip tích hợp khác và Vtel muốn đặt càng nhiều chíp của họ lên mainboard càng tốt. Chip của Vtel có dạng hình chữ nhật kích thước bằng 2x3 ô vuông của mainboard và tất nhiên chúng ta có thể đặt ngang hoặc dọc theo các chiều của mainboard.

Hãy viết chương trình giúp công ty Vtel tính xem trên một mainboard có thể đặt bao nhiêu chip của họ.

### Input

Dữ liệu của chương trình được cho trong 1 file text như sau: dòng đầu là một số nguyên D tương ứng với số test trong file (1 ≤ D ≤ 5). Mỗi test bắt đầu bằng một dòng ghi 3 số N, M và K (1 ≤ N ≤ 150, 1 ≤ M ≤ 10, 1 ≤ K ≤ M\*N) trong đó N, M là kích thước của mainboard và K là số ô đã có các chip khác của mainboard. Tiếp theo là K dòng mỗi dòng là vị trí của một ô đã có chip của mainboard (các hàng và các cột của mainboard đánh số từ 1).

### Output

Ghi kết quả xử lý của chương trình vào một file text, mỗi test trên 1 dòng riêng biệt.

### Ví dụ

|  |  |
| --- | --- |
| **Input** | **Output** |
| **2**  **6 6 5**  **1 4**  **4 6**  **2 2**  **3 6**  **6 4**  **6 5 4**  **3 3**  **6 1**  **6 2**  **6 4** | **3**  **4** |

## Bài 6: Buy low, buy lower

Mr Momus là một người rất ham mê chơi cổ phiếu trên thị trường chứng khoán và chiến thuật mà ông ta thực hiện là mua càng nhiều cổ phiếu càng tốt nhưng lần sau mua phải rẻ hơn lần trước.

Cho biết giá cổ phiếu niêm yết vào các ngày liên tiếp khác nhau trong một khoảng thời gian, hãy viết chương trình giúp Mr Momus tính xem có thể mua vào những ngày nào để có thể thực hiện phương châm mua ngày càng rẻ hơn của mình.

### Input

Dữ liệu của chương trình được cho trong một file text như sau: Dòng đầu tiên là số ngày niêm yết cổ phiếu N (1 ≤ N ≤ 5000). Dòng thứ hai là N số nguyên tương ứng với giá cổ phiếu vào các ngày từ 1 tới N.

### Output

Ghi kết quả xử lý của chương trình là số các ngày có thể mua cổ phiếu và số các nghiệm thỏa mãn yêu cầu đề bài. Các nghiệm gồm các ngày tạo thành một dãy ký tự giống nhau được xem là một nghiệm.

### Ví dụ

|  |  |
| --- | --- |
| **Input** | **Output** |
| **12**  **68 69 54 64 68 64 70 67 78 62 98 87** | **4 2** |

## Bài 7: CompanyMessages

Trong các công ty các nhân viên hay gửi các thông điệp cho nhau bằng các phần mềm chat trên mạng chẳng hạn như Yahoo, Messenger. Việc gửi thông điệp này được mô tả bằng một ma trận vuông với phần tử ở hàng i cột j tương ứng với số thông điệp mà nhân viên thứ i gửi cho nhân viên thứ j trong công ty. Hãy viết chương trình xếp thứ hạng cho các nhân viên trong công ty dựa vào số thông điệp gửi đi của họ. Việc xếp hạng này cần tiến hành sao cho số lượng thông điệp gửi từ người có thứ hạng thấp hơn cho người có thứ hạng cao hơn là nhỏ nhất. Nếu có nhiều nghiệm hãy đưa ra nghiệm có phần tử đầu tiên nhỏ nhất, nếu hai nghiệm có chu phần tử đầu tiên thì đưa ra nghiệm có phần tử thứ hai nhỏ nhất, vân vân.

### Input

Dữ liệu của chương trình được cho trong một file text, dòng đầu tiên là số nguyên ghi kích thước của ma trận (nhỏ hơn 15), tiếp theo là các hàng ghi các số của ma trận. Các phần tử của ma trận được viết liền nhau và chỉ nhận các giá trị từ 0 tới 9.

### Output

Ghi bảng xếp hạng của công ty vào một file text.

### Ví dụ

|  |  |
| --- | --- |
| **Input** | **Output** |
| **3**  **000**  **100**  **010** | **2 1 0** |
| **6**  **012105**  **601745**  **830479**  **043086**  **349904**  **649810** | **4 2 1 5 0 3** |

## Bài 8: Distincter

Cho một dãy các số nguyên. Với mỗi thao tác chúng ta có thể tăng hoặc giảm một phần tử đi một đơn vị. Hãy viết chương trình tính xem cần tối thiểu thực hiện bao nhiêu thao tác để dãy có ít nhất K phần tử khác nhau.

### Input

Dữ liệu của chương trình được cho trong một file text như sau: Dòng đầu là 2 số nguyên N, K trong đó N là số phần tử của dãy (1 ≤ N ≤ 50, 1 ≤ K ≤ N). Sau đó là dòng ghi N số nguyên (mỗi số đều nhỏ hơn 1000).

### Output

Kết quả xử lý của chương trình ghi vào một file text.

### Ví dụ

|  |  |
| --- | --- |
| **Input** | **Output** |
| **3 2**  **5 1 3** | **0** |
| **7 5**  **1 1 1 1 1 1 1** | **6** |
| **7 6**  **1 1 1 1 1 2 3** | **6** |

## Bài 9: DivisiblePermutations

Chúng ta định nghĩa một hoán vị của một số nguyên N là một số nguyên chứa các chữ số giống như N nhưng theo một thứ tự khác. Hai hoán vị của N được xem là khác nhau nếu như số mà chúng biểu diễn là khác nhau. Chẳng hạn đối với số N = 313 chúng ta có các hoán vị của N là {133, 313, 331}.

Hãy viết chương trình xác định xem với hai số nguyên N (có tối đa 15 chữ số), M (1 ≤ M ≤ 50) cho trước có bao nhiêu hoán vị của N chia hết cho M.

### Input

Dữ liệu của chương trình được cho trong một file text, có thể gồm nhiều test, mỗi test gồm 2 số nguyên được ghi trên 2 dòng riêng biệt.

### Output

Với mỗi test hãy ghi kết quả xử lý của chương trình vào một dòng của file output.

### Ví dụ

|  |  |
| --- | --- |
| **Input** | **Output** |
| **133**  **7** | **1** |
| **2753**  **5** | **6** |
| **1112225**  **5** | **20** |
| **123456789**  **17** | **21271** |

## Bài 10: Weights and Measures

Đoàn xiếc MagicX có một tiếc mục cân bằng với các nghệ sĩ rùa xếp chồng lên mai nhau. Các chú rùa có trọng lượng khác nhau và cũng có khả năng chịu đựng khác nhau nên không thể xếp đặt các chú lên nhau một cách cẩu thả.

Hãy viết một chương trình xác định xem có thể xếp tối đa bao nhiêu chú rùa biểu diễn tiết mục cân bằng.

### Input

Dữ liệu của chương trình được cho trong file text gồm các mô tả về các chú rùa, mỗi chú trên 1 dòng với 2 số nguyên tượng ứng là trọng lượng và khả năng chịu đựng tính theo gram.

### Output

Hãy ghi kết quả tìm được vào 1 file text.

### Ví dụ

|  |  |
| --- | --- |
| **Input** | **Output** |
| **300 1000**  **1000 1200**  **200 600**  **100 101** | **3** |

## Bài 11: Autoput

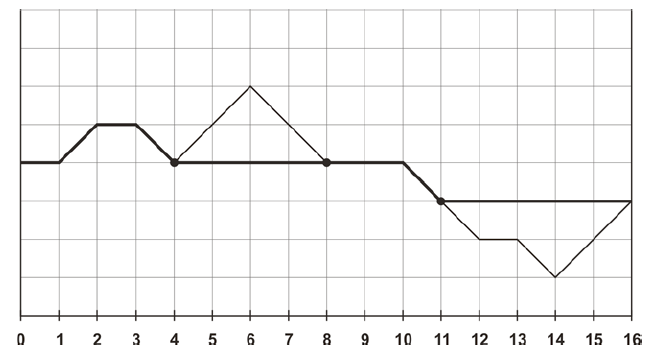
Hãy tưởng tượng một con đường trên một hệ trục tọa độ đi từ trái qua phải, trong khoảng 1 ô vuông của trục tọa độ (coi như có đơn vị đo là 1) con đường có thể:

1. giữ nguyên khoảng cách
2. đi xuống hoặc đi lên 1 ô.

Ô tô đi theo con đườngtừ trái qua phải, qua mỗi ô thời gian đi của ô tô sẽ là A giây nếu như con đường có kiến trúc như trường hợp a và B giây nếu là trường hợp b.

Tuy nhiên chúng ta có thể xây một đường hầm đi qua núi hoặc một cầu vượt qua các thung lũng. Các cầu vượt và hầm ngầm này buộc phải là đường ngang (song song với trục hoành) và thời gian đi qua mỗi ô trên đường hầm hoặc cầu vượt là C giây.

Hãy viết chương trình xác định thời gian ngắn nhất để đi hết con đường với điều kiện cho phép xây không quá K cầu vượt và đường hầm.



Trên đây là minh họa cho giải pháp không xây quá 2 cầu vượt và đường hầm.

### Input

Dòng đầu tiên của file input chứa 3 số nguyên A, B, C (1 ≤ A, B, C ≤ 100). Dòng thứ hai chứa hai số nguyên N và K (1 ≤ N ≤ 100000, 1 ≤ K ≤ 300). Dòng thứ 3 của file input chứa một dãy N ký tự mô tả về con đường từ trái qua phải trong đó các ký tự có thể là:

‘D’: có nghĩa là ô tiếp theo sẽ đi xuống

‘R’: có nghĩa là ô tiếp theo sẽ giữ nguyên độ cao

‘G’: nghĩa là ô tiếp theo sẽ đi lên

### Output

Ghi kết quả vào 1 file text.

### Ví dụ

|  |  |
| --- | --- |
| **Input** | **Output** |
| **3 2 1**  **9 1**  **GGDGGDDRR** | **16** |
| **3 5 4**  **10 10**  **RGDRDRRRRG** | **36** |
| **10 20 15**  **16 2**  **RGRDGGDDRRDDRDGG** | **235** |

## Bài 12: StringReplacement

Cho một xâu ký tự chỉ gồm 1 ký tự. Sau mỗi giây tất cả các ký tự ‘a’ của xâu sẽ được thay thế bằng chuỗi “acb”, tất cả các ký tự ‘b’ sẽ được thay bằng xâu “bba”, tất cả các ký tự ‘c’ sẽ được thay bằng xâu “bcb”. Việc thay thế này diễn ra đồng thời với tất cả các ký tự trong xâu. Chẳng hạn nếu ban đầu xâu là “a” thì sau 2 giây sẽ chuyển thành “acbbcbbaa” và giữa hai vị trí 2 và 6 sẽ là 0 ký tự a, 4 ký tự b và 1 ký tự c. Hãy viết chương trình tính xem sau N giây thì giữa 2 vị trí left và right sẽ có bao nhiêu ký tự a, bao nhiêu ký tự b và bao nhiêu ký tự c (các ký tự trong xâu được đánh số từ 0).

### Input

Dữ liệu của bài toán được cho trong một file text như sau: dòng đầu là xâu khởi tạo với 1 ký tự. 3 dòng sau là các số nguyên left, right và N. (0 ≤ N ≤ 20, 0 ≤ right ≤ 3N, 0 ≤ left ≤ right).

### Output

Kết quả của chương trình ghi vào 1 file text.

### Ví dụ

|  |  |
| --- | --- |
| **Input** | **Output** |
| **A**  **2**  **6**  **2** | **0 4 1** |
| **A**  **0**  **2**  **1** | **1 1 1** |
| **C**  **1**  **4**  **2** | **2 1 1** |

## Bài 13: FibonacciSum

Hãy xem các phần tử của dãy Fibonacci:

1, 1, 2, 3, 5, 8, 13, 21, 34, …

Tất cả các số này đều là tổng của hai số đứng ngay trước nó trong dãy. Người ta đã chứng minh được rằng một số nguyên dương bất kỳ luôn là tổng của một số các số thuộc dãy Fibonacci (các số này không nhất thiết khác nhau). Hãy viết chương trình xác định số ít nhất các số của dãy Fibonacci có tổng bằng N.

### Input

Dữ liệu input là một số nguyên N (1 ≤ N ≤ 1000000).

### Output

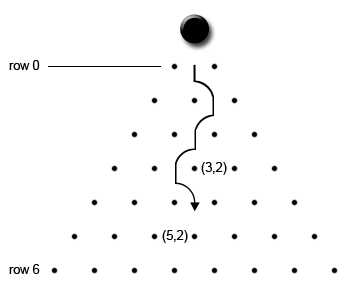
Đưa ra số ít nhất các số có tổng bằng N.

### Ví dụ

|  |  |
| --- | --- |
| **Input** | **Output** |
| **1** | **1** |
| **7** | **2** |
| **70** | **3** |
| **107** | **3** |

## Bài 14: FallingBall

Trong trò chơi thả bóng một quả bóng rơi từ đỉnh của một hình tam giác xuống đáy. Hình tam giác được chia thành các ô và có n hàng. Quả bóng sẽ rơi xuống theo từng hàng cho tới khi nó chạm vào hàng cuối cùng. Trên mỗi hàng quả bóng có thể rơi sang trái hoặc sang phải. Hàng đầu tiên chỉ có 1 ô, hàng thứ hai có 2 ô, vân vân. Có thể hình dung qua hình vẽ sau (các chổ trống chính là các ô).



Các hàng được đánh số từ đỉnh tới đáy và bắt đầu từ 0, trên các cột các ô được đánh số bằng đầu từ 0 từ trái qua phải. Chú ý là hàng i sẽ có i+1 ô được đánh số từ 0 tới i và nếu quả bóng nằm trên ô k của hàng i nó sẽ rơi xuống ô k ở hàng i+1 nếu sang trái và ô k+1 trên hàng i+1 nếu rơi sang phải.

Cho một danh sách các ô (không quá 50) và kích thước của bảng trò chơi là N (1 ≤ N ≤ 30) hãy tính xem có bao nhiêu cách quả bóng có thể rơi từ trên đỉnh xuống hàng dưới cùng của bảng trò chơi đi qua các ô trong danh sách.

### Input

Dữ liệu của chương trình được cho trong một file text như sau: ban đầu là danh sách các ô được cho bởi 2 số nguyên/1 ô trên mỗi dòng, sau đó là một dòng chỉ chứa số nguyên N.

### Output

Ghi kết quả xử lý của chương trình vào một file text.

### Ví dụ

|  |  |
| --- | --- |
| **Input** | **Output** |
| **3 2**  **5 2**  **7** | **6** |
| **0 0**  **0 9**  **30** | **536870912** |
| **0 0**  **29 0**  **30** | **1** |
| **10 0**  **10 1**  **15** | **0** |

## Bài 15: FixedSizeSums

Cho hai số nguyên sum và count. Có nhiều cách để biểu diễn sum dưới dạng tổng của count số nguyên dương. Hai cách sẽ được xem là một nếu như cách này là một hoán vị của cách kia. Chẳng hạn ta có các cách sau 8 = 3 + 2 + 1 + 1 + 1, 8 = 1 + 3 + 2 + 1 + 1 được tính là một cách nhưng 8 = 3 + 2 + 2 + 1 lại là một cách khác. Trong các cách biểu diễn khác nhau đó chún ta chỉ quan tâm tới cách biểu diễn mà các số tham gia vào biểu diễn đó tạo thành một dãy không tăng.

Chẳng hạn với sum = 8 và count = 3 chúng ta sẽ có các cách biểu diễn sau:

8 = 6+1+1

8 = 3+3+2

8 = 4+2+2

8 = 4+3+1

8 = 5+2+1

Chúng ta có thể sắp thứ tự cách các biểu diễn này bằng cách so sánh các biểu diễn từ trái qua phải gặp phần tử đầu tiên khác nhau của hai biểu diễn, nếu số của biểu diễn nào lớn hơn thì sẽ đứng trước. Chẳng hạn với cách đánh thứ tự như vậy các dãy biểu diễn trên sẽ xuất hiện theo thứ tự sau:

8 = 6+1+1

8 = 5+2+1

8 = 4+3+1

8 = 4+2+2

8 = 3+3+2

Viết chương trình nhập vào 3 số nguyên sum, count và index hãy đưa ra biểu diễn thứ index của sum với count số nguyên theo cách tính thứ tự như trên (1 ≤ sum, count ≤ 150, index < 2000000000).

### Input

Dữ liệu của chương trình được cho trong 1 file text với 3 số ghi trên 3 dòng riêng biệt.

### Output

Ghi kết quả xử lý vào một file text.

### Ví dụ

|  |  |
| --- | --- |
| **Input** | **Output** |
| **8**  **3**  **2** | **8 = 4 + 3 + 1** |
| **13**  **1**  **0** | **13** |
| **5**  **5**  **0** | **5 = 1 + 1 + 1 + 1 + 1** |
| **7**  **10**  **0** |  |

## Bài 16: Home on the Range

Mr John chăn thả đàn súc vật của anh trên trên một cánh đồng hình vuông rộng lớn có độ dài các cạnh là N dặm và vì một số lý do nào đó đàn gia súc của John chỉ ăn cỏ trên các vùng đất hình vuông. Cánh đồng được chia nhỏ thành các ô vuông kích thước 1 dặm và có một số ô không có cỏ. Hãy viết chương trình giúp Mr John tính xem diện tích của các khu còn lại (không tính các khu có diện tích là 1x1) có thể chăn thả là bao nhiêu (tìm các khu vực hình vuông chứa cỏ), và với mỗi diện tích đó thì có bao nhiêu khu vực như vậy (tất nhiên các khu vực này có thể chồng lên nhau).

### Input

Dữ liệu của chương trình được cho như sau: dòng đầu chứa số nguyên N ( 2 ≤ N ≤ 250) là kích thước của cánh đồng. Sau đó là N dòng mỗi dòng gồm N số nguyên hoặc 0 hoặc 1 viết liền nhau, 0 có nghĩa là không có cỏ còn 1 có nghĩa là có cỏ.

### Output

Kết quả cần sắp theo thứ tự tăng dần của các diện tích của các khu vực tìm được và ghi vào 1 file text.

### Ví dụ

|  |  |
| --- | --- |
| **Input** | **Output** |
| **6**  **101111**  **001111**  **111111**  **001111**  **101101**  **111001** | **2 10**  **3 4**  **4 1** |

## Bài 17: NumberSplit

Bắt đầu trò chơi chia số người ta chia các chữ số của một số nguyên thành ít nhất 2 phần (là các số nguyên) sau đó lấy các phần đó nhân lại với nhau để tạo thành một số mới, lặp lại quá trình đó với số mới cho tới khi số mới thu được chỉ có một chữ số. Chẳng hạn đối với số 234 chúng ta có thể có các cách chia như sau:

 23 \* 4 = 92,

 2 \* 34 = 68 và

 2 \* 3 \* 4 = 24.

Nếu chúng ta chọn cách chia thứ 2 thì số 68 sẽ là số tiếp theo và sẽ chia thành 6\*8 = 48, 4\*8 = 32, 3\*2 = 6. Theo cách này chúng ta tạo thành một dãy có 5 phần tử (234, 68, 48, 32, 6). Theo hai cách còn lại chúng ta có thể tạo thành các dãy (234, 92, 18, 8) và (234, 24, 8) đều có ít phần tử hơn theo cách thứ 2 ban đầu.

Hãy viết chương trình tính xem có thể tạo thành một dãy dài nhất là bao nhiêu số với một số ban đầu theo qui luật chia trên.

### Input

Dữ liệu input là một số nguyên không quá 100000. Chú ý là bài toán luôn có nghiệm và chúng ta có thể bỏ qua các số không đứng đầu trong quá trình chia chẳng hạn 3021 = 3\*021 = 63.

### Output

Ghi kết quả tìm được vào một file text.

### Ví dụ

|  |  |
| --- | --- |
| **Input** | **Output** |
| **4**  **2 4 1**  **3 1 2 4**  **1 2**  **4** | **Co the truy cap**  **1 2 4**  **Khong the truy cap**  **3** |

## Bài 18: Money Systems

Cho V loại tiền với các mệnh giá khác nhau hãy viết chương trình tính xem có bao nhiêu cách đổi N đồng tiền thành các đồng tiền khác trong số V loại tiền đã cho.

### Input

Dữ liệu của chương trình được cho trong một file text như sau: dòng đầu là 2 số nguyên V, N (1 ≤ V ≤ 25, 1 ≤ N ≤ 10,000). Dòng thứ hai là V số nguyên tương ứng với các mệnh giá của V loại tiền.

### Output

Kết quả xử lý của chương trình ghi vào một file text.

### Ví dụ

|  |  |
| --- | --- |
| **Input** | **Output** |
| **3 10**  **1 2 5** | **10** |

## Bài 19: New Album

Nhóm nhạc Symphony X đã quyết định tung ra thị trường một album nhạc Rock mới của họ, Album này có N bài hát. Tất cả các bài hát đều có thời gian bằng nhau là L giây. Một đĩa CD hiện nay có thể ghi được C giây. Hai bài hát liền nhau đòi hỏi mất 1 giây cho đoạn chuyển tiếp từ bài này sang bài khác. Giám đốc sản xuất cho nhóm là một người rất mê tín và ông ta quyết định không có đĩa CD nào chứa số bài hát chia hết cho 13. Hãy viết chương trình tính xem cần sử dụng ít nhất bao nhiêu đĩa CD trắng để ghi hết số bài hát trong album.

### Input

Dữ liệu của chương trình được cho trong một file text gồm 3 số nguyên ghi trên 3 dòng riêng biệt. Đầu tiên là N (1 ≤ N ≤ 100), dòng thứ hai là C (1 ≤ C ≤ 10000) và dòng thứ ba là L (1 ≤ L ≤ C).

### Output

Kết quả của chương trình ghi vào một file text.

### Ví dụ

|  |  |
| --- | --- |
| **Input** | **Output** |
| **7**  **2**  **6** | **4** |
| **20**  **1**  **100** | **1** |
| **26**  **1**  **100** | **2** |
| **26**  **3**  **51** | **3** |

## Bài 20: Nule

Cho một bảng trò chơi hình vuông gồm NxN ô vuông, mỗi ô chứa một số nguyên không âm. Khi bắt đầu trò chơi một mảnh nhựa nhỏ được đặt tại ô (1, 1) của bảng trò chơi và nhiệm vụ của người chơi là phải di chuyển nó tới ô (N, N). Mỗi thao tác di chuyển chỉ có thể di chuyển xuống dưới hoặc sang phải 1 ô và không được di chuyển vào ô chứa số 0. Chúng ta định nghĩa chi phí của một con đường đi từ ô đầu tiên tới ô cuối cùng của trò chơi là tích của tất cả các số trên các ô đã đi qua. Và một con đường sẽ được coi là tối ưu nếu như số lượng các số không ở cuối chi phí của nó là nhỏ nhất.

Hãy viết chương trình xác định số lượng số không của đường đi tối ưu của trò chơi.

### Input

Dữ liệu của chương trình được cho trong một file text như sau: dòng đầu là một số nguyên N (1 ≤ N ≤ 1000). Sau đó là N dòng, mỗi dòng gồm N số. Các số trong bảng trò chơi không quá 1000000. Chú ý là bài toán luôn có nghiệm

### Output

Kết quả xử lý của chương trình ghi vào một file text.

### Ví dụ

|  |  |
| --- | --- |
| **Input** | **Output** |
| **3**  **1 2 3**  **4 5 6**  **7 8 9** | **0** |
| **3**  **5 7 6**  **4 0 1**  **3 2 5** | **1** |
| **4**  **1 3 0 0**  **0 8 2 25**  **6 5 0 3**  **0 15 7 4** | **2** |

## Bài 21: Palindromist

Một xâu ảnh gương là một xâu đối xứng và có kết quả như nhau dù chúng ta đọc xuôi hay ngược xâu. Cho một nửa xâu của một xâu ảnh gương, hãy đưa ra xâu ảnh gương đầy đủ được ghép bằng các từ liên tiếp lấy từ một tập các từ cho trước. Xâu kết quả phải được tách thành các từ riêng biệt. Chú ý là với mỗi nửa xâu chúng ta có thể có 2 xâu ảnh gương có thể là kết quả của bài toán, ví dụ nửa xâu cho là “ABC” thì xâu kết qủa có thể là “ABCCBA” hoặc “ABCBA”. Nửa xâu ban đầu được cho dưới dạng một xâu liền không có dấu cách giữa các ký tự.

### Input

Dữ liệu của bài toán được cho trong một file text như sau: dòng đầu là nửa xâu của xâu ảnh gương cần tìm, sau dòng thứ hai là danh sách các từ cho trước, mỗi từ cách nhau một dấu cách.

### Output

Kết quả xử lý của chương trình ghi vào một file text.

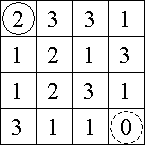
### Ví dụ

|  |  |
| --- | --- |
| **Input** | **Output** |
| **AMANAPLANAC**  **A CANAL MAN PANAMA PLAN** | **A MAN A PLAN A CANAL PANAMA** |
| **AAAAA**  **AA A AAA** | **A A A A A A A A A** |
| **RACEFAST**  **AR CAR FAST RACE SAFE CEFA ACE STTS AFEC** | **RACE FAST SAFE CAR** |

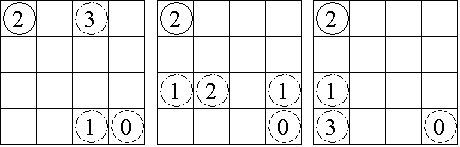
## Bài 22: Pascal’s Travel

Trên các ô của một bảng vuông kích thước NxN người ta đặt các số nguyên không âm. Mục đích của trò chơi là xuất phát từ ô góc trên bên trái (1, 1) của bảng xuống tới vị trí của ô phía dưới bên phải (N, N). Số nguyên trong mỗi ô là khoảng cách di chuyển từ ô đó tới vị trí tiếp theo theo hướng sang trái hoặc đi xuống. Số 0 được đặt ở ô đích.

Chúng ta có thể xem 2 hình minh họa sau:



Hình1: Bảng trò chơi



Hình 2: Các cách di chuyển

Hãy viết chương trình tính xem có bao nhiêu con đường có thể có.

### Input

Dữ liệu của chương trình được cho trong một file text, có thể gồm nhiều test. Mỗi test bắt đầu bằng một dòng ghi 1 số nguyên là kích thước của bảng trò chơi N ( 4 ≤ N ≤ 34) sau đó là N dòng, mỗi dòng gồm N số nguyên không âm nhỏ hơn 10 viết liên tiếp nhau. Số test trong file nhiều nhất là 13 test. Kết thúc file input là một dòng ghi số -1

### Output

Kết quả mỗi test ghi trên 1 dòng.

### Ví dụ

|  |  |
| --- | --- |
| **Input** | **Output** |
| **4 2331 1213 1231 3110 4 3332 1213 1232 2120 5 11101 01111 11111 11101 11101 -1** | **3 0 7** |

## Bài 23: PieSharing

Có 3N chiếc bánh trên bàn xếp thành một vòng tròn. Tom và Jerry sẽ ăn hết bánh theo qui luật sau: Tom có thể chọn bất cứ chiếc bánh nào để ăn nhưng sau đó Jerry sẽ ăn hai chiếc bánh kề ngay bên cạnh chiếc bánh Tom đã ăn. Sau N lần số bánh sẽ được ăn hết. Kích thước của các chiếc bánh là khác nhau. Hãy viết chương trình giúp Tom tính xem có thể ăn được tối đa số bánh (tính theo tổng kích thước của các bánh đã ăn) là bao nhiêu.

### Input

Dữ liệu của chương trình được cho trong một file text như sau: kích thước của các chiếc bánh là các số nguyên từ 1 tới 100 được ghi trên 1 dòng ngăn cách với nhau bằng một dấu cách. Số bánh luôn là một số nguyên chia hết cho 3 và không quá 48.

### Output

Kết quả xử lý của chương trình ghi vào một file text.

### Ví dụ

|  |  |
| --- | --- |
| **Input** | **Output** |
| **33 33 34** | **34** |
| **5 17 22 34 18 4** | **51** |
| **11 1 1 85 1 1** | **96** |
| **6 13 14 4 14 10 1 20 18** | **48** |

## Bài 24: Pimp My Ride

Có N công việc được đánh số từ 1 tới N. Mỗi công việc sẽ phải trả p USD và với mỗi cặp công việc (i, j) (i ≠ j) cần phải trả thêm s đồng nếu chúng ta tiến hành công việc j trước i. Hãy viết chương trình tính xem tổng số tiền cần trả ít nhất là bao nhiêu để hoàn thành N công việc.

### Input

Dữ liệu input của chương trình được cho trong file text như sau: dòng đầu tiên là số test trong file, sau đó mỗi test sẽ bắt đầu bằng một dòng ghi số công việc N ( 1 ≤ N ≤ 14), tiếp đến là N dòng, mỗi dòng gồm N số nguyên trong đó số nguyên ở cột i, hàng i là giá cho công việc i còn các số khác chính là số tiền phải trả thêm nếu công việc j tiến hành trước công việc i. Các số nguyên tương ứng với số tiền phải trả nằm trong khoảng từ 1 tới 100000.

### Output

Với mỗi test cần in số hiệu test và số tiền phải trả ít nhất.

### Ví dụ

|  |  |
| --- | --- |
| **Input** | **Output** |
| **2**  **2**  **10 10**  **9000 10**  **3**  **14 23 0**  **0 14 0**  **1000 9500 14** | **Test 1: 30**  **Test 2: 42** |

## Bài 25: Score Inflation

Trong một cuộc thi lập trình online người ta không hạn chế số lượng bài làm nhưng khống chế trong một khoảng thời gian nhất định. Các bài tập được chia thành N nhóm, mỗi nhóm với số lượng không hạn chế. Mỗi bài trong một nhóm chỉ được làm trong một khoảng thời gian nhất định. Hãy viết chương trình tính xem một lập trình viên tham gia cuộc thi có thể đạt được tối đa bao nhiêu điểm.

### Input

Dữ liệu của chương trình được cho trong một file text như sau: dòng đầu là hai số nguyên M, N (1 ≤ M, N ≤ 10000) trong đó M là thời gian hạn chế của cuộc thi còn N là số nhóm bài thi. Tiếp theo là N dòng, mỗi dòng gồm 2 số tương ứng là điểm và thời gian hạn chế của các bài trong N nhóm bài thi tương ứng.

### Output

Kết quả xử lý của chương trình ghi vào một file text.

### Ví dụ

|  |  |
| --- | --- |
| **Input** | **Output** |
| **300 4**  **100 60**  **250 120**  **120 100**  **35 20** | **605**  **(Làm 2 bài thi nhóm 2 và 3 bài thi nhóm 4)** |

## Bài 26: ShortTaps

Giả sử chúng ta cần gửi một số thông điệp có độ dài khác nhau (tính về mặt thời gian) và không muốn số thông điệp bị chặn lại vượt quá 3. Một thông điệp sẽ được xem là bị chặn lại nếu như một người xấu bụng nào đó khóa kết nối trong suốt thời gian mà đáng lẽ thông điệp phải được gửi đi (việc khóa cục bộ không làm thông điệp bị chặn lại). Giả sử kết nối bị khóa lại trong khoảng thời gian liên tiếp **interceptTime** phút, thì thời gian ngắn nhất cần thiết để chúng ta gửi tất cả các thông điệp đi mà số thông điệp bị chặn lại không vượt quá 3 sẽ là bao nhiêu.

Mỗi thông điệp được gửi đi trong một truyền thông liên tiếp cho phép gửi các thông điệp khác một cách song song. Chúng ta chỉ có thể bắt đầu gửi một thông điệp tại thời điểm bắt đầu của 1 phút. Các thông điệp có thể gửi theo bất cứ thứ tự nào.

### Input

Dữ liệu của chương trình được cho trong một file text như sau: dòng đầu là thời gian đường truyền sẽ bị khóa **interceptTime** (1 ≤ **interceptTime** ≤ 100), dòng thứ hai là các số nguyên tương ứng với thời gian gửi (độ dài thời gian) các thông điệp cách nhau bởi 1 dấu cách (thời gian gửi 1 thông điệp không quá 50 phút).

### Output

Kết quả xử lý của chương trình ghi vào 1 file text.

### Ví dụ

|  |  |
| --- | --- |
| **Input** | **Output** |
| **10**  **2 3 4 5 6 7 8** | **14** |
| **20**  **14 2 9 14 17 1 3 10 5 9 25 8 11 7** | **43** |
| **40**  **30 40 50 60 70 80 90 100** | **100** |

Chú ý: nghiệm đầu tiên của bài toán tương ứng với đáp án sau:

**Time Message length**

**---------------------**

**0 2 min, 3 min, 6 min**

**3 8 min**

**6 5 min**

**7 4 min, 7 min**

## Bài 27: Trip

Tom và Jerry đều muốn đi du lịch nhân dịp ngày nghỉ lễ. Cả hai đều có kế hoạch riêng gồm các thành phố sẽ đi thăm trên đường du lịch theo một thứ tự nhất định.

Tuy nhiên nếu đi một mình thì cũng không thú vị lắm nên cả hai quyết định sẽ đi cùng nhau trên cùng một hành trình. Mặc dù đi chung nhưng cả hai đều không muốn thay đổi thứ tự của các thành phố sẽ đến nên chỉ có 1 giải pháp là bỏ đi một số thành phố trong hành trình của mỗi người sao cho hành trình chung sẽ càng dài càng tốt.

Hãy viết chương trình giúp Tom và Jerry tìm tất cả các hành trình dài nhất có thể được để hai người có thể lựa chọn.

### Input

Tên các thành phố là các chữ cái thường từ a tới z. Dữ liệu input của bài toán được cho trong một file text gồm 2 dòng tương ứng là hành trình hai bên đã lựa chọn trước. Mỗi hành trình là danh sách tên các thành phố viết liên tiếp nhau. Mỗi danh sách không nhiều hơn 80 thành phố.

### Output

Kết quả của chương trình ghi vào một file text, mỗi hành trình trên một dòng riêng biệt. Chú ý là bài toán luôn có ít nhất 1 nghiệm và không có nhiều hơn 1000 nghiệm.

### Ví dụ

|  |  |
| --- | --- |
| **Input** | **Output** |
| **4**  **2 4 1**  **3 1 2 4**  **1 2**  **4** | **Co the truy cap**  **1 2 4**  **Khong the truy cap**  **3** |

## Bài 28: Sum of Consecutive Prime Numbers

Một số số nguyên có thể biểu diễn dưới dạng tổng của một hoặc nhiều số nguyên tố liên tiếp. Chẳng hạn 53 có thể biểu diễn thành 5 + 7 + 11 + 13 + 17 hoặc 53. 41 có 3 cách biểu diễn khác nhau là: 2 + 3 + 5 + 7 + 11 + 13 = 11 + 13 + 17 = 41. Chú ý là các biểu diễn trên sử dụng các số nguyên tố liên tiếp và không lặp lại.

Hãy viết chương trình tính xem một số nguyên bất kỳ có thể có bao nhiêu cách biểu diễn như trên.

### Input

Dữ liệu của chương trình cho trong một file text gồm các số nguyên (nhỏ hơn 10000 và lớn hơn 1) viết trên các dòng riêng biệt, kết thúc file input là 1 dòng ghi số 0.

### Output

Với mỗi số nguyên trong file input hãy đưa ra số cách biểu diễn tương ứng trên 1 dòng của file output.

### Ví dụ

|  |  |
| --- | --- |
| **Input** | **Output** |
| **2**  **3**  **17**  **41**  **20**  **666**  **12**  **53**  **0** | **1**  **1**  **2**  **3**  **0**  **0**  **1**  **2** |

## Bài 29: Stringsobits

Nhập vào 3 số nguyên dương N L và I. Hãy in ra xâu bit thứ I trong dãy các xâu bit có độ dài N (N bit) mà số bit 1 trong các xâu bít này không vượt quá L.

### Input

Dữ liệu của chương trình được cho trong một file text gồm 3 số nguyên N L I trên 1 dòng ( 1 ≤ N ≤ 31 1 ≤ L ≤ N 1 ≤ I ≤ sizeof(tập các xâu N bit thỏa mãn điều kiện đầu bài)).

### Output

Ghi kết quả xử lý của chương trình vào một file text.

### Ví dụ

|  |  |
| --- | --- |
| **Input** | **Output** |
| **5 3 19** | **10011** |

## Bài 30: Spreading Gossip

Tại một ngôi làng ở một vùng xa xôi hẻo lánh có N hộ gia đình (N ≤ 20) h0, h1, …, hN và một số đường liên lạc điện thoại nối giữa các ngôi nhà là hàng xóm của nhau (chính xác là có một đường liên lạc giữa mỗi cặp 2 nhà là hàng xóm của nhau). Giữa hai căn hộ hi và hj bất kỳ luôn có ít nhất 1 đường điện thoại (có thể xem như đây là một đồ thị vô hướng liên thông với đỉnh là các căn hộ và các cạnh là các đường điện thoại).

Các câu chuyện phiếm có thể được lan truyền qua hệ thống điện thoại. Mỗi căn hộ có thể gọi tới nhiều nhất 1 căn hộ hàng xóng tại một thời điểm. Mỗi lần gọi điện chỉ có thể thực hiện vào thời điểm bắt đầu của một tiếng (chẳng hạn như 9 am, 1 pm, 6 pm vân vân và kéo dài chính xác trong 1 tiếng.

Hệ thống viễn thông tính cước các cuộc gọi cục bộ dựa trên mỗi lần gọi nhưng có một điểm lợi là tất cả các cuộc gọi được tiến hành đồng thời chỉ được tính cước bằng 1 cuộc gọi.

Hãy tìm chi phí nhỏ nhất (số lượng thời gian nhỏ nhất) để phổ biến một câu chuyện từ căn hộ h0 tới tất cả các căn hộ khác.

### Input

File input có thể gồm nhiều test. Mỗi test bắt đầu bằng một dòng ghi số N, sau đó là N dòng, mỗi dòng ghi chỉ số của các căn hộ tương ứng là hàng xóm của căn hộ hi (tương ứng với dòng thứ i sau dòng đầu tiên) và các chỉ số này không tuân theo một thứ tự nhất định nào. Kết thúc file input là test có N = 0.

### Output

Kết quả mỗi test cần ghi trên 1 dòng riêng biệt của file output.

### Ví dụ

|  |  |
| --- | --- |
| **Input** | **Output** |
| **4**  **1 2**  **0 3**  **3 0**  **1 2**  **7**  **1 2 3**  **0 2**  **0 1 3 4**  **0 2**  **6 2 5**  **4 6**  **4 5**  **0** | **Test 1: 2**  **Test 2: 4** |

## Bài 31: Mobile sevice

Một công ty cung cấp các dịch vụ cho các đối tác có trụ sở tại các thành phố khác nhau. Công ty có 3 nhân viên hỗ trợ dịch vụ di chuyển để phục vụ các khách hàng. Nếu như có một yêu cầu tại một thành phố nào đó một nhân viên sẽ phải di chuyển từ vị trí hiện tại tới nơi khách hàng yêu cầu (trong trường hợp không có nhân viên nào đang ở đó). Chỉ có một nhân viên có thể thực hiện việc di chuyển trong một thời điểm bất kỳ. Các nhân viên chỉ di chuyển vì công việc và không được phép ở cùng một vị trí. Việc di chuyển từ vị trí p tới vị trí q sẽ đòi hỏi một chi phí bằng C(p, q). Chi phí không nhất thiết là đối xứng và tất nhiên không thể bằng 0. Nguyên tắc phục vụ của công ty đối với các yêu cầu của khách hàng là first-come, first-service. Mục đích của công ty là giảm thiểu chi phí cho việc đáp ứng các yêu cầu của khách hàng.

Hãy viết chương trình tính xem đối với một danh sách các yêu cầu phục vụ cho trước công ty cần phải trả tối thiểu bao nhiêu tiền cho việc di chuyển của các nhân viên của mình.

### Input

Dòng đầu tiên của file input chứa hai số nguyên L và N (3 ≤ L ≤ 200, 1 ≤ N ≤ 1000) trong đó L là số thành phố và N là số yêu cầu phục vụ. Các thành phố được đánh số từ 1 tới L. Tiếp theo là L dòng, mỗi dòng chứa L số nguyên không âm trong đó số thứ j trên dòng thứ i+1 là chi phí C(i, j) (C(i, j) ≤ 2000). Dòng cuối cùng chứa N số nguyên là danh sách các yêu cầu. Mỗi yêu cầu là tên thành phố tương ứng nơi nhân viên cần đến phục vụ. Tại thời điểm ban đầu ba nhân viên của công ty đang ở tại các thành phố 1, 2 và 3.

### Output

Dòng đầu của file output chứa tổng chi phí nhỏ nhất công ty cần trả M, dòng thứ hai là N số nguyên tương ứng với thứ tự phục vụ của các yêu cầu, mỗi phục vụ hoặc là 1, 2 hoặc 3 có nghĩa là nhân viên 1, 2 hoặc 3 sẽ đáp ứng yêu cầu phục vụ đó.

### Ví dụ

|  |  |
| --- | --- |
| **Input** | **Output** |
| **5 9**  **0 1 1 1 1**  **1 0 2 3 2**  **1 1 0 4 1**  **2 1 5 0 1**  **4 2 3 4 0**  **4 2 4 1 5 4 3 2 1** | **5**  **1 2 1 2 2 1 3 1 3** |

# IV. Greedy

# V. Other