1. เบาสุด (lightest)

มีของหล่นมาจากฟากฟ้า หล่นมาในหมวกวิเศษ ของแต่ละชิ้นมีน้ำหนักและมูลค่าต่าง ๆ ระหว่างที่ของหล่นมา เราอาจหยิบของในหมวกวิเศษนี้ ของที่เราหยิบได้จะเป็นของที่มีน้ำหนักน้อยที่สุดที่มีในหมวกขณะนั้น

ให้เขียนโปรแกรมเพื่อหาลำดับของมูลค่าของของที่หยิบได้ทั้งหมด

ข้อมูลป้อนเข้า

บรรทัดแรกมีจำนวนเต็ม N และ M (1<= N <=100,000; 1<= M <=100,000) แทนจำนวนของและจำนวนครั้งของการหยิบของ จากนั้นอีก N + M บรรทัดจะประกอบด้วยข้อมูลของของหรือคำสั่งการหยิบของ โดยแต่ละบรรทัดมีรูปแบบดังนี้

ถ้าบรรทัดดังกล่าวขึ้นต้นด้วยอักษร 'T' จะเป็นการระบุว่ามีของหล่นลงมาในหมวก จำนวนเต็ม W V (1<= W <= 1,000,000,000; 1<= V <=10,000) ที่ตามมา จะระบุน้ำหนัก และมูลค่าของของชิ้นนั้น

ถ้าบรรทัดดังกล่าวขึ้นต้นด้วยอักษร 'P' จะเป็นการระบุว่าเราจะหยิบของจากหมวกในขณะนั้น

จำนวนบรรทัดที่ขึ้นต้นด้วยอักษร T จะเท่ากับ N และจำนวนบรรทัดที่ขึ้นต้นด้วยอักษร P จะเท่ากับ M

ผลลัพธ์

มี M บรรทัด แต่ละบรรทัดระบุมูลค่าของของที่หยิบได้ในการหยิบครั้งต่าง ๆ ถ้าในขณะนั้นไม่มีของให้หยิบให้พิมพ์ 0

ตัวอย่าง

input:

```
3 4
T 10 20
T 5 10
P
T 7 30
P
P
P
```

output:

```
10
30
20
0
```

2. ซ้ายขวา (bstpractice)

คุณได้รับข้อมูลเป็นจำนวนเต็มที่ไม่ซ้ำกันเพื่อนำมาจัดเก็บด้วย binary search tree ในข้อนี้เราจะให้คุณเขียนผลลัพธ์ที่ได้จากการวิ่งเพิ่มโหนดในต้นไม้

ในข้อนี้ให้สร้าง bst แบบมาตรฐาน ไม่ต้องกังวลเกี่ยวกับความลึกหรือเวลาการทำงาน

ข้อมูลป้อนเข้า

บรรทัดแรกมีจำนวนเต็ม N (1 <= N <= 100,000) จากนั้นอีก N บรรทัดจะระบุจำนวนเต็มบรรทัดละจำนวนที่จะเพิ่มเข้าไปใน binary search tree จำนวนเต็มแต่ละจำนวนมีค่าระหว่าง -1,000,000,000 ถึง 1,000,000,000

ผลลัพธ์

มี N บรรทัด แสดงเส้นทางการวิ่งไปในต้นไม้เพื่อที่จะเพิ่มโหนดแต่ละโหนด ให้พิมพ์ไล่ไปตามลำดับ ถ้าต้องท่องไปทางโหนดด้านซ้ายพิมพ์ L ถ้าต้องท่องไปทางโหนดด้านขวาให้พิมพ์ R เมื่อหยุด (ถึงจุดหมาย) ให้พิมพ์ *

ตัวอย่าง

input:

```
7
1
2
5
4
3
-2
-1
```

output:

```
*
R*
RR*
RRL*
RRLL*
L*
LR*
```

3. ขวาซ้าย (tracebst)

คุณได้สังเกตโปรแกรมของเพื่อนที่รับข้อมูลเป็นจำนวนเต็มที่ไม่ซ้ำกันเพื่อนำมาจัดเก็บด้วย binary search tree อย่างไรก็ตามคุณไม่เห็นข้อมูลป้อนเข้าของโปรแกรมนั้น สิ่งที่คุณเห็นคือเส้นทางการท่องไปในต้นไม้ของโปรแกรมดังกล่าว ในรูปแบบของข้อ 2 เช่น

```
*
R*
RR*
RRL*
RRLL*
L*
LR*
```

หลังจากได้รับข้อมูลดังกล่าว คุณต้องการสร้างลำดับของข้อมูลป้อนเข้าที่เป็นจำนวนเต็มบวก ที่ไม่ซ้ำกัน โดยมีเงื่อนไขว่าเมื่อนำข้อมูลดังกล่าวไปป้อนให้กับโปรแกรมของเพื่อนคุณแล้ว จะทำให้โปรแกรมมีเส้นทางการเพิ่มข้อมูลเหมือนกับที่คุณได้รับมา

รับประกันว่ามีข้อมูลจริง ที่ทำให้โปรแกรมดังกล่าวมีเส้นทางการท่องไปในต้นไม้เหมือนกับที่ระบุ

ข้อมูลป้อนเข้า

บรรทัดแรกมีจำนวนเต็ม N (1 <= N <= 100,000) จากนั้นในอีก N บรรทัดจะระบุเส้นทาง กล่าวคือ ในบรรทัด i จะระบุสตริง S_i แทนเส้นทางการท่องไปในต้นไม้เพื่อเพิ่มข้อมูลแต่ละตัวในรูปแบบข้างต้น รับประกันว่าไฟล์อินพุตที่มีขนาดใหญ่ที่สุดจะมีขนาดไม่เกิน 3 MB

ผลลัพธ์

มี N รายการจำนวนเต็มบวกที่ทำให้โปรแกรมมีเส้นทางการท่องไปในต้นไม้ตามที่ระบุ โดยจำนวนเต็มบวกแต่ละตัวจะต้องมีค่าระหว่าง 1 ถึง N

ตัวอย่าง

input:

```
7
*
R*
RR*
RRL*
RRLL*
L*
LR*
```

output:

```
3
4
7
6
5
1
2
```

4. ขวาซ้ายซ้ำ (tracebst2)

คุณได้สังเกตโปรแกรมของเพื่อนที่รับข้อมูลเป็นจำนวนเต็มที่อาจซ้ำกันได้เพื่อนำมาจัดเก็บด้วย binary search tree สำหรับกรณีที่ข้อมูลที่นำไปเพิ่มซ้ำกับข้อมูลเดิม โปรแกรมจะวิ่งไปจนพบแล้วหยุดโดยไม่มีการเพิ่มข้อมูล

เช่นเดียวกับข้อที่แล้ว คุณไม่เห็นข้อมูลป้อนเข้าของโปรแกรมนั้น สิ่งที่คุณเห็นคือเส้นทางการท่องไปในต้นไม้ของโปรแกรมดังกล่าว ในรูปแบบของข้อ 2 เช่น

```
*
R*
RR*
RRL*
RRLL*
RR*
L*
LR*
```

หลังจากได้รับข้อมูลดังกล่าว คุณต้องการสร้างลำดับของข้อมูลป้อนเข้าที่เป็นจำนวนเต็มบวก โดยมีเงื่อนไขว่าเมื่อนำข้อมูลดังกล่าวไปป้อนให้กับโปรแกรมของเพื่อนคุณแล้ว จะทำให้โปรแกรมมีเส้นทางการเพิ่มข้อมูลเหมือนกับที่คุณได้รับมา

รับประกันว่ามีข้อมูลจริง ที่ทำให้โปรแกรมดังกล่าวมีเส้นทางการท่องไปในต้นไม้เหมือนกับที่ระบุ

ข้อมูลป้อนเข้า

บรรทัดแรกมีจำนวนเต็ม N (1 <= N <= 100,000) จากนั้นในอีก N บรรทัดจะระบุเส้นทาง กล่าวคือ ในบรรทัด i จะระบุสตริง S_i แทนเส้นทางการท่องไปในต้นไม้เพื่อเพิ่มข้อมูลแต่ละตัวในรูปแบบข้างต้น รับประกันว่าไฟล์อินพุตที่มีขนาดใหญ่ที่สุดจะมีขนาดไม่เกิน 3 MB

ผลลัพธ์

มี N บรรทัด เป็นรายการจำนวนเต็มบวกที่ทำให้โปรแกรมมีเส้นทางการท่องไปในต้นไม้ตามที่ระบุ โดยจำนวนเต็มบวกแต่ละตัวจะต้องมีค่าระหว่าง 1 ถึง M เมื่อ M คือจำนวนของจำนวนเต็มบวกที่แตกต่างกันในจำนวนเต็มบวก N ตัวนั้น

ตัวอย่าง

input:

```
8
*
R*
RR*
RRL*
RRLL*
RR*
L*
LR*
```

output:

```
3
4
7
6
5
7
1
2
```