



少年圖靈計畫
Young Turing Program

0_送分題 (Hello World)

(30分)

時間限制: 1 second

記憶體限制: 256 MB

前言

比賽開始了！

趕快驗證一下，

網路是否設定正確？

上傳競賽程式是否順利？

程式解答是否用 STDOUT 輸出？

都沒問題，30分就到手了！繼續 ... 衝！衝！衝！

題目敘述

請寫一個程式輸出Hello World!

輸入格式

本題無需輸入值

輸出格式

[A~Z][a~z]、空格，以及常用英文符號。

資料範圍

[A~Z][a~z]、空格，以及驚嘆號“!”

測試範例

輸入範例 1

(無輸入值)

輸出範例 1

```
Hello world!
```

範例說明

輸入範例1, 無輸入值，簡單而快樂的輸出Hello World!

1_要是某天能變成大人的話就好了呢 (I_Hope_You_Can_Be_an_Adult_Someday)

(10分)

時間限制: 1 second

記憶體限制: 256 MB

題目敘述

「果然還是下雨了呢。」

被比較的孩子看著彷彿要下雨的模樣，倒數著距離變成大人還要經過多少個日子。

「要是某天能變成大人的話就好了呢。」

你知道變成大人不是經過了 20 年就會自動完成的事情，不過你還是可以幫他算算，距離變成 20 歲還要經過多少年的時間。或告訴他，他已經滿 20 歲了，不再是個小朋友了。

輸入格式

輸入只有一個整數，表示被比較的孩子的年齡 X 。

輸出格式

輸出一個整數，如果被比較的孩子的年齡已經是 20 歲或以上了，請輸出 -1 ；否則，請輸出他還差幾歲才 20 歲。

資料範圍

- $0 \leq X \leq 100$

測試範例

輸入範例 1

19

輸出範例 1

1

輸入範例 2

20

輸出範例 2

-1

輸入範例 3

30

輸出範例 3

-1

範例說明

輸入範例 1 中，被比較的孩子還差 1 歲才 20 歲。

輸入範例 2 和 3 中，被比較的孩子已經滿 20 歲了，故輸出 -1。

2_炸彈 (Bomb)

(10 分)

時間限制: 1 second

記憶體限制: 256 MB

題目敘述

蓉蓉是一位帶領著 12 位社員的熱舞社社長，為了能讓社員們快速建立感情，他在第一堂課就教大家玩了一種刺激的團康遊戲，規則如下：

所有社員圍成一個圈，並依照順時針編號（1 號 ~ 12 號）。社長在圓心點當莊家，莊家會從撲克牌堆中依照編號由小到大幫所有人頭上綁一張牌，以至於每個人都不知道自己頭上綁著的牌是什麼。此外，莊家還會紀錄每個人的積分，初始為 1000。

遊戲開始時會輪流喊撲克牌上的數字，1 號喊 1，2 號喊 2 ... 12 號喊 12，1 號喊 13，2 號重新從 1 開始喊，以此類推，如果有人喊到跟牌一樣的數字就輸了，並且會有莊家指定的懲罰。懲罰完後，莊家會在他頭上綁上一張新的牌，然後把他的積分扣除新的牌的點數，接著開啟新的一回合，從輸家的下一位繼續，從 1 開始喊，直到 m 回合後結束。

這遊戲之所以刺激，是因為輸家不知道自己被扣了多少點，只有在遊戲結束公布積分時才會知道。因為莊家只有社長一人，她擔心自己的計算錯誤會對其他人不公平，因此想請你幫忙計算所有人的總積分以確保無誤。

註：每次綁新牌時，莊家都會從撲克牌堆中拿取最上面的牌。

輸入格式

第一行有 52 個整數 a_i ，以空白隔開，表示撲克牌堆從上到下的排列順序（每個數字皆有 4 張）。

第二行有一個整數 m ，表示社長想要進行的回合數。

輸出格式

輸出一個整數，表示所有社員的積分總和。

資料範圍

- $1 \leq a_i \leq 13$ 。
- $1 \leq m \leq 40$ 。

測試範例

輸入範例 1

讀取以下測資時，請特別注意輸入格式的說明。

因紙本顯示的行寬有限，以下3筆輸入範例的第一行有部分資料轉折顯示於第二行。（看起來像在第二行，實際上仍是第一行的資料！）

```
6 2 9 1 3 9 1 2 5 1 6 13 13 3 9 13 12 4 7 10 11 5 7 8 10 1 9 4 11 4 8 12 12 11 6 7 10 12
11 5 5 6 3 3 13 8 7 8 4 2 10 2
1
```

輸出範例 1

```
11987
```

輸入範例 2

```
6 2 9 1 3 9 1 2 5 1 6 13 13 3 9 13 12 4 7 10 11 5 7 8 10 1 9 4 11 4 8 12 12 11 6 7 10 12
11 5 5 6 3 3 13 8 7 8 4 2 10 2
2
```

輸出範例 2

```
11984
```

輸入範例 3

```
6 2 9 1 3 9 1 2 5 1 6 13 13 3 9 13 12 4 7 10 11 5 7 8 10 1 9 4 11 4 8 12 12 11 6 7 10 12
11 5 5 6 3 3 13 8 7 8 4 2 10 2
40
```

輸出範例 3

```
11694
```

範例說明

在範例 1 中，社長依序從牌堆中拿出 12 張牌並綁在社員頭上，此時社員頭上的牌依序為 [6 2 9 1 3 9 1 2 5 1 6 13]。遊戲開始時，一號喊 1，二號喊 2，此時正好喊到跟頭上一樣的數字，因此二號社員須接受懲罰。懲罰完後，莊家會從牌堆中拿一張牌 (13) 並綁在二號社員的頭上，接著把積分版上二號社員的積分扣 13，因此遊戲結束時的積分為：

- 2 號：987
- 其餘人：1000

總和為 11987。

在範例 2 中，接續了範例 1 的狀況，因為二號輸了，所以從三號重新開始，三號喊 1，四號喊 2，五號喊 3，此時又喊到跟頭上相同的數字，因此五號需接受懲罰。懲罰完後，莊家會從牌堆中拿一張牌 (3) 並綁在五號社員的頭上，接著把積分版上五號社員的積分扣 3，因此遊戲結束時的積分為：

- 2號：987
- 5號：997
- 其餘人：1000

總和為 11984。

3_蛋餅愛質數(Omelet_Loves_Prime_Number)

(15 分)

時間限制: 1 second

記憶體限制: 256 MB

題目敘述

蛋餅有 N 個數字 $a_1, a_2, \dots, a_N$ ，這些數字兩兩相異。因為蛋餅很喜歡質數，他想要從這裡面挑選一些數字使兩兩相異數字的總和都是質數，你能幫幫他計算總共有幾種方法嗎？

更明確的說，請計算有幾個序列 $b_1, b_2, \dots, b_K$ 滿足：

- $1 \leq b_1 < b_2 < b_3 < \dots < b_K \leq N$
- 對於所有 $1 \leq i < j \leq K$ ， $a_{b_i} + a_{b_j}$ 是質數

輸入格式

輸入第一行有一個正整數 N ，代表數字的數量。

輸入第二行有 N 個正整數 $a_1, a_2, \dots, a_N$ ， a_i 代表第 i 個數字為何。

輸出格式

請輸出一個整數，代表有幾種序列滿足條件。保證在資料範圍的限制下，答案會在 64 bit 整數範圍內。

資料範圍

- $2 \leq N \leq 60$
- $1 \leq a_i \leq 1000$
- $a_i \neq a_j \forall 1 \leq i < j \leq N$

測試範例

輸入範例 1

```
3
3 1 4
```

輸出範例 1

```
5
```

輸入範例 2

```
5
2 4 6 8 10
```

輸出範例 2

```
5
```

範例說明

在第一筆範例測資中，總共有以下五種可能的選法：

- [3]
- [1]
- [4]
- [3, 4]
- [1, 4]

4_電梯 (Elevators)

(3 分 / 3 分 / 9 分)

時間限制: 2 seconds

記憶體限制: 256 MB

題目敘述

當我們著手規劃新建科技大樓時，其中的電梯系統設計顯然是一項關鍵性的考量。這裡所討論的電梯系統包括了每部電梯的停靠樓層範圍和移動時間等重要資訊，這些資訊的安排將直接影響到大樓內部的交通效率和使用者的舒適度。

透過合理的電梯設計，我們可以有效地縮短人們從大樓一樓到達特定樓層的時間。這對於提升大樓的整體運作效率和提供更優質的使用體驗至關重要。

現在，我們正著手規劃一棟高為 H 的大樓，裡面有 N 部電梯，每部電梯有三個資訊 d_i, u_i, t_i ，代表該電梯可以從 d_i 移動到 u_i 層樓（也就是閉區間 $[d_i, u_i]$ ，包含端點），且移動一層樓需要 t_i 秒，在不考慮轉換電梯所需要的移動和等待時間，請問從一樓到 H 的最短時間為多少秒？如果無法從一樓到 H 樓，請輸出 -1 。

輸入格式

輸入的第一行包含兩個正整數 H 、 N ，代表大樓的高度、電梯數量。

接下來有 N 行輸入，每行有三個正整數 d_i, u_i, t_i ，如題意所述。

輸出格式

輸出一個正整數，代表所需要的最短時間，如果無法到達請輸出 -1 。

資料範圍

- $1 \leq H \leq 10^9$ 。
- $1 \leq N \leq 3 \times 10^5$ 。
- $1 \leq d_i \leq u_i \leq H$ 。
- $1 \leq t_i \leq 10^3$ 。

子任務

- 子任務 1 滿足 $1 \leq H \leq 10^5$ 、 $1 \leq N \leq 10^3$ 。
- 子任務 2 滿足 $1 \leq H \leq 3 \times 10^5$ 、 $1 \leq N \leq 3 \times 10^5$ 。
- 子任務 3 無額外限制。

測試範例

輸入範例 1

```
50 3
20 50 5
35 50 2
1 30 3
```

輸出範例 1

```
142
```

輸入範例 2

```
50 3
20 40 5
45 50 2
1 30 3
```

輸出範例 2

```
-1
```

輸入範例 3

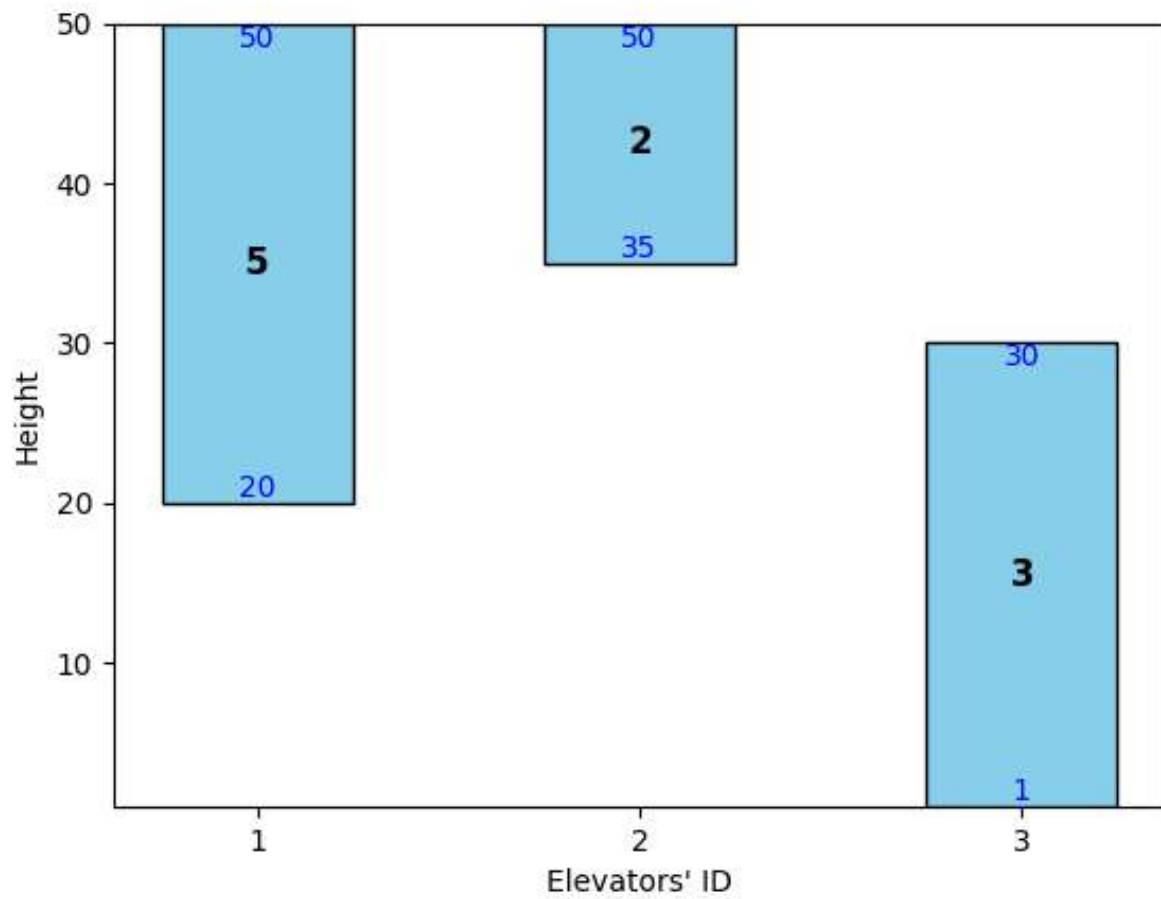
```
50 7
45 50 5
32 48 7
23 40 6
1 28 10
10 35 8
37 50 12
1 13 5
```

輸出範例 3

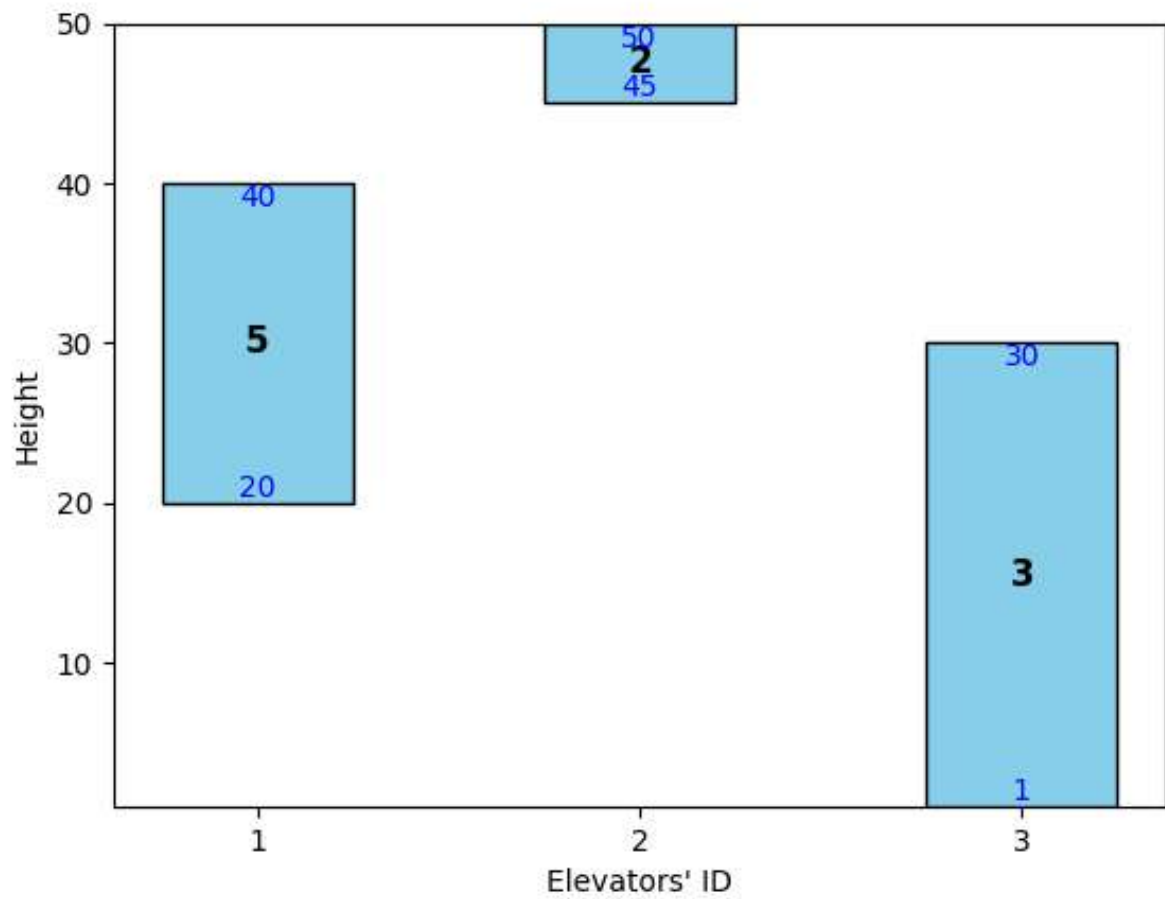
```
302
```

範例說明

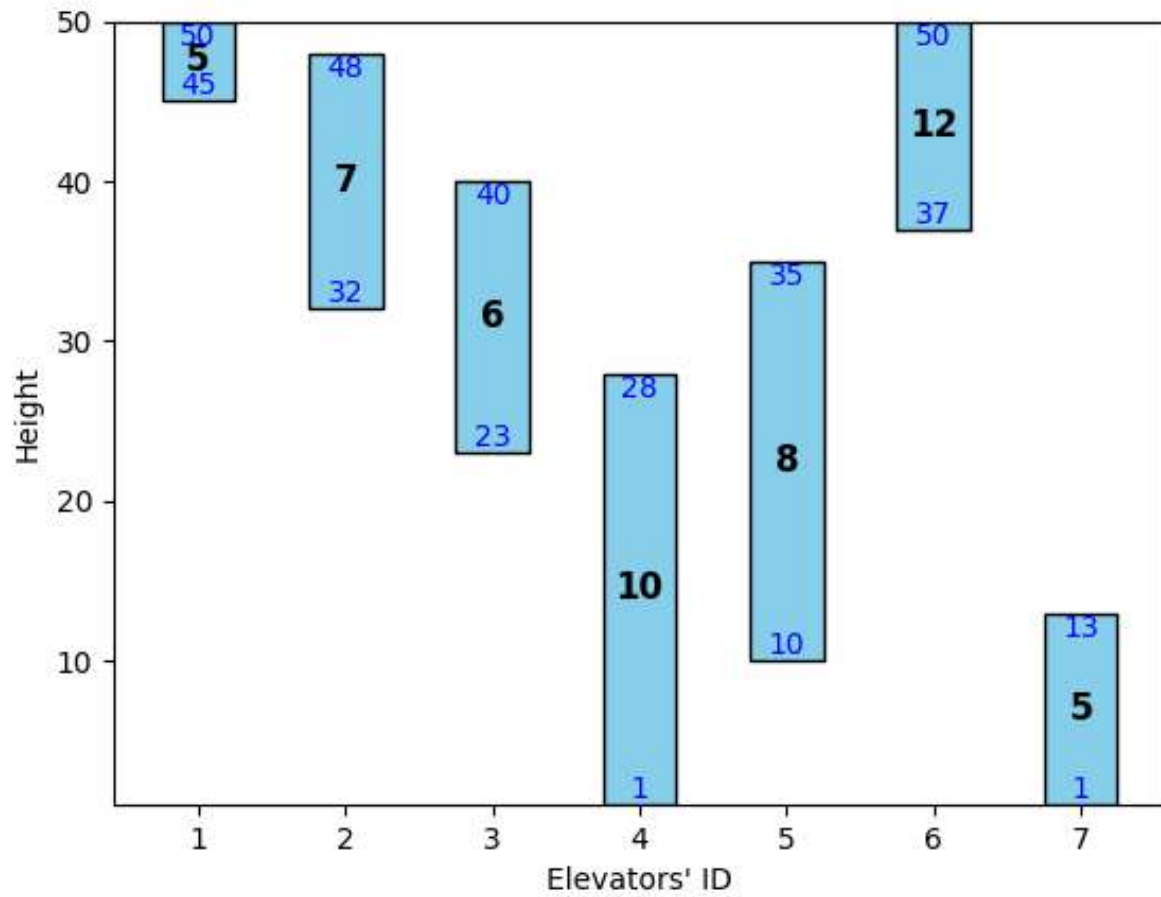
在範例 1 中，1 ~ 30 樓可以搭第三部電梯，30 ~ 35 樓可以搭第一部電梯，35 ~ 50 樓可以搭第二部電梯，總共花的時間為 $29 \times 3 + 5 \times 5 + 15 \times 2 = 112$ 。



在範例 2 中，40 ~ 45 樓沒有電梯可以搭，因此沒有辦法從一樓搭到 H 樓，所以輸出 -1 。



在範例 3 中，1 ~ 13 樓可以搭第七部電梯，13 ~ 23 樓可以搭第五部電梯，23 ~ 40 樓可以搭第三部電梯，40 ~ 45 樓可以搭第二部電梯，45 ~ 50 樓可以搭第一部電梯，總共花的時間為 $12 \times 5 + 10 \times 8 + 17 \times 6 + 5 \times 7 + 5 \times 5 = 302$ 。



5_合成斧頭(Crafting_Axes)

(15 分)

時間限制: 1 second

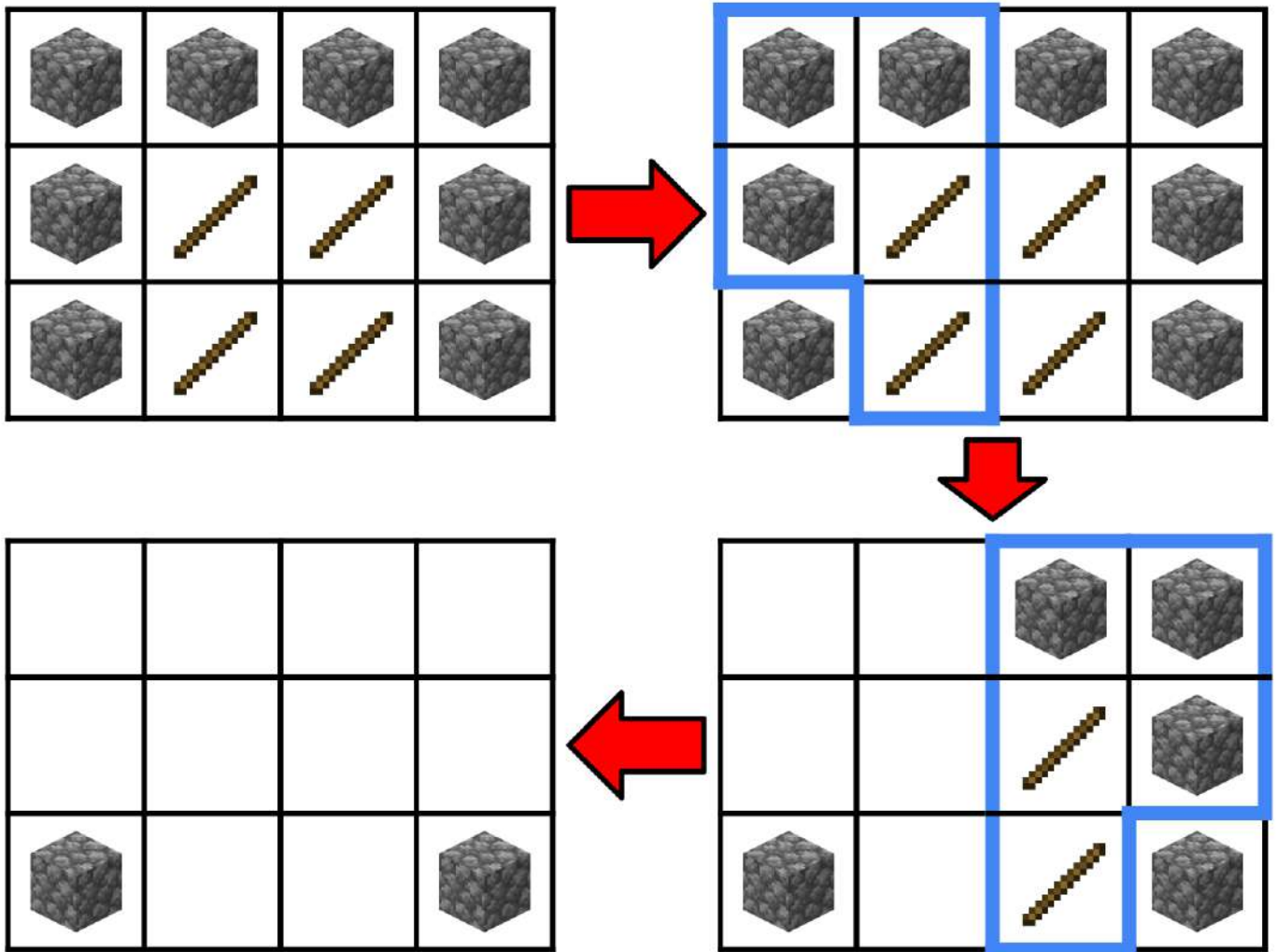
記憶體限制: 256 MB

題目敘述

小方塊喜歡玩方塊遊戲。他有一個 $N \times M$ 的表格，每一格都會放著石頭或是木棍，小方塊想要用這個表格合成出越多個石斧越好。令第 i 橫列的第 j 格為 (i, j) ，每次小方塊可以進行以下兩種操作之一：

- 挑選一個格子 (i, j) 滿足 $1 \leq i \leq N - 3, 1 \leq j \leq M - 2$ ，且 $(i, j), (i + 1, j), (i, j + 1)$ 格子放著石頭， $(i + 1, j + 1), (i + 2, j + 1)$ 放著木棍。小方塊可以用這些格子合成出一個石斧，合成完後這些位置都會變成空氣。
- 挑選一個格子 (i, j) 滿足 $1 \leq i \leq N - 3, 1 \leq j \leq M - 2$ ，且 $(i, j), (i, j + 1), (i + 1, j + 1)$ 格子放著石頭， $(i + 1, j), (i + 2, j)$ 放著木棍。小方塊可以用這些格子合成出一個石斧，合成完後這些位置都會變成空氣。

舉例來說，下圖為一個合成的例子：



現在給定表格的內容，請幫小方塊計算他最多可以合成出幾個石斧。

輸入格式

輸入第一行有兩個正整數 N, M ，代表表格的大小。

接下來 N 行，第 i 行有一個長度為 M 個字串 s_i ，若 $s_{i,j} = 0$ 則代表格子 (i, j) 放的是石頭，否則代表為木棍。

輸出格式

請輸出一行，該行有一個整數，代表最多可以合成出幾個石斧。

資料範圍

- $1 \leq N, M \leq 500$
- $|s_i| = M$
- $s_{i,j} \in \{0, 1\}$

測試範例

輸入範例 1

```
3 4
0000
0110
0110
```

輸出範例 1

```
2
```

輸入範例 2

```
5 5
00000
00000
00000
00000
00000
```

輸出範例 2

```
0
```

輸入範例 3

```
4 4
0000
0100
0110
1110
```

輸出範例 3

```
2
```

範例說明

題目敘述中的圖說明了第一筆範例測資，答案為 2。

在第二筆範例測資中，因為沒有木棍所以答案顯然為 0。

6_超大杯珍奶(Bubblelee_Loves_Bubble_Tea)

(15分)

時間限制: 1 second

記憶體限制: 256 MB

題目敘述

小圓村是一個座落於丘陵上的村莊，這個看似平凡的村莊其實隱藏著一個驚為天人的祕密。那就是這個村子的地底下，其實隱藏著一個由珍珠奶茶所組成的地下湖！這個地下珍珠奶茶湖供應著整個村莊的飲水需求。據統計，小圓村的平均每個村民一天都會飲用 1.5 公升的珍珠奶茶。居民們每天早上起來都會拿著一個 2 公升的量筒到地下珍珠奶茶湖旁邊裝一杯超大杯珍奶作為一天的飲用水。

BubbleLee 從小便在小圓村長大，身為小圓村的一員，他最喜歡喝珍珠奶茶了！不過，為了工作，他不得不離開有著地下珍珠奶茶湖的小圓村，前往大方鎮為生活打拼。沒想到，在大方鎮，他發現這裡又有許多不同販賣珍珠奶茶的店家！他興奮的買了 N 杯不同店家的珍珠奶茶，回到租屋處準備享用。

這時，他卻發現，這些珍珠奶茶的珍奶比相差太大了。珍奶比是指一杯珍珠奶茶中，珍珠對奶茶的比例（即 $\frac{\text{珍珠量}}{\text{奶茶量}}$ ）。身為珍珠奶茶的愛好者，BubbleLee 有著自己的原則，就是他不能接受一次喝的 M 杯珍珠奶茶當中，有其中 2 杯珍珠奶茶的珍奶比差大於 K 。現在他想要最大化能喝到的奶茶量，請問若他在不違反原則的情況下從 N 杯珍珠奶茶中選一些來喝，則他最多可以喝多少單位的奶茶？

輸入格式

第一行包含三個正整數 N, x, y ，代表接下來有 N 杯珍珠奶茶的資訊，而 x, y 則代表 $K = \frac{x}{y}$ 。

接下來 N 行，每行包含兩個正整數 b_i, m_i ，代表第 i 杯珍珠奶茶的珍珠量和奶茶量。

輸出格式

請輸出 BubbleLee 最多能喝到多少單位的奶茶。

資料範圍

- $1 \leq N \leq 2 \times 10^5$
- $1 \leq x, y, b_i, m_i \leq 10^6$

測試範例

輸入範例 1

```
5 1 4
4 5
5 10
3 3
7 8
7 9
```

輸出範例 1

```
25
```

輸入範例 2

```
10 1 10
200 1500
100 1000
150 2000
70 500
80 700
50 1600
600 2500
202 143
2061 411
2161 411
```

輸出範例 2

```
5700
```

範例說明

在範例 1 中，我們可以選擇喝第 1, 3, 4, 5 杯珍奶，這樣總共喝了 $5 + 3 + 8 + 9 = 25$ 單位的奶茶。

在範例 2 中，我們可以選擇喝第 1, 2, 3, 4, 5 杯珍奶，這樣總共喝了 $1500 + 1000 + 2000 + 500 + 700 = 5700$ 單位的奶茶。

7_過馬路(Traversing_Crossroads)

(20分)

時間限制: 1 second

記憶體限制: 512 MB

題目敘述

「怎麼辦，快要趕不上比賽了！」參加今年 YTP 的你，現在快要遲到了，因此你需要盡快的從捷運站走到比賽會場。

捷運站到比賽會場中間的路由 N 條東西向道路和 M 條南北向道路交錯而成。我們將從北方數第 i 條東西向道路和從西方數第 j 條南北向道路的交叉路口用 (i, j) 表示。捷運站在 $(1, 1)$ ，比賽會場在 (N, M) 。

任兩條相鄰東西向道路和任兩條相鄰南北向道路的距離都需要你走一分鐘。

這裡的交通號誌有點特別，可以用一個字串 S 表示紅燈的狀態：如果 $S[i]$ 是 **W**，代表東西向的所有路口在第 i 分鐘都是紅燈，其他則是綠燈；如果 $S[i]$ 是 **H**，代表南北向的所有路口在第 i 分鐘都是紅燈，其他則是綠燈；如果 $S[i]$ 是 **.**，代表所有路口都是綠燈。你到捷運站的時間是第 1 分鐘。

在這 NM 個路口中，有一些路口不能走。而因為你很趕時間，你不想花任何時間在等紅燈或繞路，所以每一分鐘你都要從 (i, j) 走到 $(i + 1, j)$ 或 $(i, j + 1)$ 。走到一半的時候你開始想，有多少種符合這些條件的解法呢？由於路徑數可能很多，請輸出答案模 $10^9 + 7$ 的結果。

註：對於兩個整數 a, b 且 $b > 0$ ， a 模 b 指的是 a 除以 b 的餘數。也可以寫成最小的非負整數 r ，使得 $a = bq + r$ 。

輸入格式

第一行有兩個整數 N, M 。

第二行有一個長度為 $N + M - 2$ 的字串 S ，可以證明在不等紅燈的情況下，最快的路徑一定需要走 $N + M - 2$ 分鐘。

接下來 N 行每一行有 M 個數字，其中 $a_{i,j}$ 為第 i 行的第 j 個數字。如果 $a_{i,j} = 0$ 代表路口 (i, j) 可以走，如果 $a_{i,j} = 1$ 代表路口不能走。

輸出格式

輸出一個整數，代表符合條件的路徑數模 $10^9 + 7$ 的結果。

子任務

- 本題只有一個子任務

資料範圍

- $1 \leq N, M \leq 1000$
- $S[i] \in \{H, W, .\}, \forall 1 \leq i \leq N + M - 2$
- $a_{i,j} \in \{0, 1\}, a_{1,1} = 0, a_{N,M} = 0$

測試範例

輸入範例 1

```
4 4
.....
0 0 0 0
0 0 1 0
0 0 1 0
1 0 0 0
```

輸出範例 1

```
4
```

輸入範例 2

```
4 4
.H.W..
0 0 0 0
0 0 1 0
0 0 1 0
1 0 0 0
```

輸出範例 2

```
2
```

範例說明

範例測試 2 的兩種走法如下（路徑經過的格子用 x 表示）

```
x 0 0 0
x x 1 0
0 x 1 0
1 x x x

x x x x
0 0 1 x
0 0 1 x
1 0 0 x
```

8_農場 (Farm)

(1 分/4 分/15 分)

時間限制: 1 second

記憶體限制: 256 MB

題目敘述

從前從前，有一個綿延數公里的農場，這整個農場可以以數線上的 x 軸表示，並且在農場中，有幾隻牛被長度不一的繩子拴在 x 軸的木樁上，使得牠們的活動範圍限制於 $[x - r, x + r]$ ，其中 x 表示木樁的位置，而 r 表示繩子的長度。每隻牛還有一個特定的飢餓值 t ，代表牠們每次進食會消耗 t 公升的食物。

由於農場主人非常懶惰，為了減輕每天餵食牛隻的疲勞，他打算在農場中建立一些自動餵食機。為了確保牛隻井然有序地進食，農場主人會先要求牛隻依照木樁所在的 x 座標依序排隊，也就是木樁座標 x 越小的牛會排在越前面，接著農場主人才會開啟自動餵食機，讓牛一隻接著一隻進食。由於位置靠右的牛會較晚獲得食物，因此牠們的飢餓程度會更高，每隻牛的飢餓值會變為自己的飢餓值加上前面排隊中所有牛隻的原始飢餓值。

因為農場主人一次要管理很多牛隻，導致他不知道要把自動餵食機設置在哪裡比較合適，因此他列了一些預定地，想請你幫他計算如果把一台自動餵食機設在該位置，至少需要準備多少公升的食物，才能滿足所有能到達自動餵食機的牛隻。

輸入格式

第一行有 2 個正整數 N 、 M ，代表牛隻數量及自動餵食機的數量。

接下來會有 N 行，每行有 3 個整數 x_i 、 r_i 、 t_i ，分別代表第 i 隻牛的 x 座標、繩子長度，以及飢餓值。

最後一行有 M 個整數 $a_1, a_2, \dots, a_M$ ，代表第 i 個自動餵食機預定地位置的 x 座標。

輸出格式

輸出一行包含 M 個整數，以空白隔開，代表該自動餵食機至少需要準備多少公升的食物，才能滿足所有能到達自動餵食機的牛隻。

資料範圍

- $1 \leq N, M \leq 2 \times 10^5$ 。
- $1 \leq x_i, r_i, t_i, a_i \leq 10^8$ 。
- $\forall i \neq j, x_i \neq x_j, a_i \neq a_j$ 。

子任務

- 子任務 1 滿足 $M \leq 10$ 。
- 子任務 2 滿足 $r_i \leq 50, x_i, a_i \leq 10^6$ 。
- 子任務 3 無額外限制。

測試範例

輸入範例 1

```
2 1
2 2 3
6 2 4
4
```

輸出範例 1

```
10
```

輸入範例 2

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

輸出範例 2

```
25
```

輸入範例 3

```
5 4
1 100 7
5 8 5
10 1 4
21 3 2
53 10 6
11 19 63 64
```

輸出範例 3

```
35 16 20 7
```

範例說明

在範例 1 中，有兩隻牛分別在 $x = 2$ 跟 $x = 6$ 的位置，繩長度都是 2，代表第 1 隻牛的活動區間為 $[0, 4]$ ，第 2 隻牛的活動區間為 $[4, 8]$ ，因此在 $x = 4$ 的自動餵食機可以餵食到兩隻牛。依照木樁所在的 x 座標排列後，第一隻牛的飢餓值為 3，第二隻牛的飢餓值為 $3 + 4 = 7$ ，因此自動餵食機至少需要準備 $3 + 7 = 10$ 公升的食物。

在範例 2 中，3 隻牛都可以到達自動餵食機，依照木樁所在的 x 座標排列後，第一隻牛的飢餓值為 5，第二隻牛的飢餓值為 $5 + 2 = 7$ ，第三隻牛的飢餓值為 $5 + 2 + 6 = 13$ ，因此自動餵食機至少需要準備 $5 + 7 + 13 = 25$ 公升的食物。

9_不要硬幣 (No_Coin)

(4 分/16 分)

時間限制: 2 seconds

記憶體限制: 1024 MB

題目敘述

小Y 是一名住在 YTP 國的平凡老百姓，每天按時工作，過著平凡而規律的生活。在第 0 天晚上，他下班回家的路上，突然在地上看到一塊閃閃發光的石子。好奇心驅使著他彎腰撿了起來，發現石子下面竟然是一張樂透彩券。

小Y 心想也許這是他的幸運日，於是急忙回家核對彩券號碼。當他對照著電視上播出的樂透開獎結果時，他不敢相信自己的眼睛-他的確中了頭獎！

興奮不已的小Y 一夜未眠，他不斷重複地看著手中的樂透彩券，確保自己並沒有做夢。這一夜，他的生活完全改變了。

兌獎後小Y 瞬間成為了百萬富翁，財產一共有 $10^{1000000}$ 元，不過一向勤儉節約的小Y 來說，他仍維持著規劃金錢支出的好習慣，他預計在接下來的 n 天內，每天花 a_i 元。然而，得到這麼多錢也造成了小Y 的困擾，其中一個最大的困擾就是零錢的問題。

YTP 國的幣值分別有 1, 5, 10, 50, 100, 1000, $10^{1000000}$ 元，其中只有 1 元的部分是硬幣。然而由於 YTP 國的金屬「鐵」價格特別便宜，硬幣皆是由密度極大的鐵製成。因此小Y 非常不喜歡持有非常重的 1 元硬幣。

知道小Y 困擾的小T 決定設計一個電子支付系統，讓小Y 能更輕易省時的付錢，然而由於小T 沒有參加過 YTP 計畫，他設計出來的系統在 n 天內最多只能使用 k 天，因此小Y 轉向有參加過 YTP 計畫的小P，請他設計一個付費方式計畫，決定每一天是否要使用現金支付或電子支付，使得每天完成支付後，剩餘硬幣的總和數量最小。

作為小P 的隊友，請你跟他一起設計出一個程式計算出最小的每天剩餘硬幣總和，以及產生最小總和時需要在哪幾天使用電子支付。

由於小Y 很討厭硬幣，因此他在付錢時永遠都會採取讓自己擁有 1 元硬幣數目最小的方法，也就是說在任意時刻，小Y 的錢包內永遠不會有超過 5 個 1 元硬幣。

輸入格式

輸入共有兩行。

第一行包含兩個整數 n, k ，分別代表總共天數以及小Y 可以使用電子支付的天數。

第二行有 n 個非負整數 $a_1, a_2, \dots, a_n$ ，代表第 i 天小Y 花錢的總金額。

輸出格式

輸出包含兩行。

第一行包含兩個非負整數 $total_cnt, days$ ，分別代表每天剩餘硬幣數量最小總和以及產生最小總和所需要使用電子支付的天數。

第二行包含 $days$ 個正整數 $ans_1, \dots, ans_{days}$ ，代表小Y 需要使用電子支付分別是哪幾天。

輸出需滿足 $0 \leq days \leq k$ 以及 ans 需是嚴格遞增數列。

如果有多種可能產生最小總和的計畫，請輸出 *days* 最小的計畫。若已最小化 *days* 仍有多解時，則輸出其中任意一個解。

資料範圍

- $1 \leq k \leq n \leq 1000$
- $0 \leq a_i \leq 10^8$

子任務

- 子任務 1 滿足 $1 \leq k \leq n \leq 20$ 。
- 子任務 2 無額外限制。

測試範例

輸入範例 1

```
5 2
1 2 3 4 5
```

輸出範例 1

```
3 2
1 4
```

輸入範例 2

```
9 1
4 2 3 3 5 5 5 5 5
```

輸出範例 2

```
5 1
2
```

輸入範例 3

```
9 3
4 2 3 3 5 15 25 35 45
```

輸出範例 3

```
3 2
1 4
```

範例說明

在範例一中，小Y 可以選擇在第 1, 4 天使用電子支付，每天剩餘的硬幣量分別是 0, 3, 0, 0, 0，因此每天剩餘硬幣量最小總和是 3。

在範例二中，小Y 可以選擇在第 2 天使用電子支付，每天剩餘的硬幣量分別是 1, 1, 3, 0, 0, 0, 0, 0, 0，因此每天剩餘硬幣量最小總和是 5。

在範例三中，小Y 可以選擇在第 1, 4 天使用電子支付，每天剩餘的硬幣量分別是 0, 3, 0, 0, 0, 0, 0, 0, 0，因此每天剩餘硬幣量最小總和是 3。請注意在這個範例中可以使用不到 k 次電子支付就達到最小總和。

10_字母辨識(Letter_Recognition)

(6分/6分/13分)

時間限制: 1 second

記憶體限制: 256 MB

題目敘述

好好吃野餐大會 (Yummy Tummy Picnic, 簡稱 YTP) 就要開始了！

為了宣傳活動，小 Y 跟小 P 列印了很多上面寫著字母 Y、T、P 的海報，每一張海報上印著其中一個字母。他們準備把這些海報交給小 T 找地方貼上，不過一走出門的瞬間，一陣強風吹來，不只吹亂了他們的頭髮，嚇壞了路邊打架的鴿子，還把這些海報吹散了一地！為了重新分類印著三種不同字母的海報，他們需要你寫一支程式來快速辨識海報上的字母，或告訴他們撿到的海報上沒有任何字母，可能只是其他人的海報。

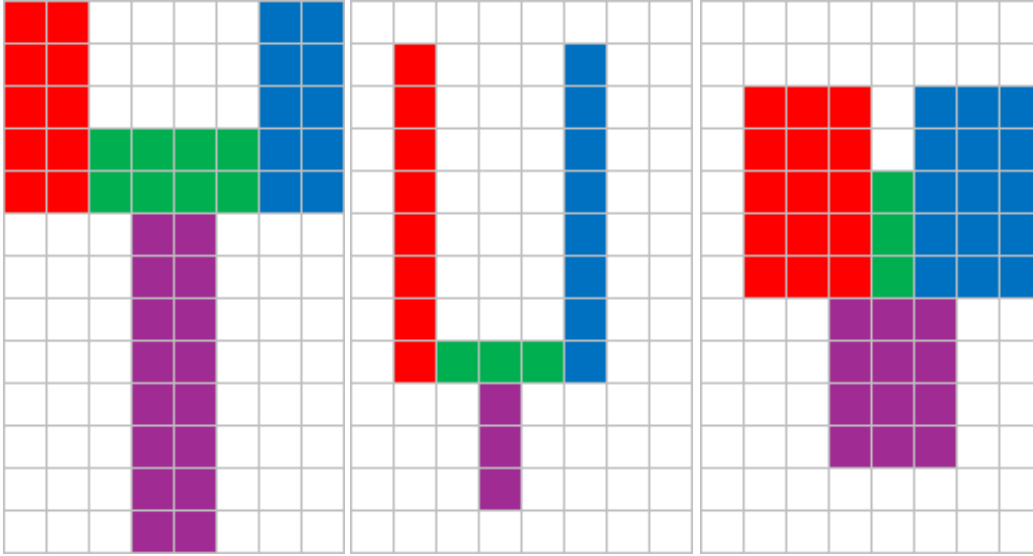
幸好這些海報上面的字體都是相同的，所以要辨識這三個字母對你來說並不困難。我們將一張海報視為 $N \times M$ 的表格，表格的每一行由上到下依序由 1 到 N 編號，每一列由左到右依序由 1 到 M 編號，並且第 i 行第 j 列的格子為 (i, j) 。每個格子可能是黑色或白色其中一種顏色。

接下來我們將定義 Y、T、P 三個字母在海報上的字體的長相。以下所提到的矩形皆為一些格子所構成的集合，對於某個矩形 X ，其左上角的格子為 (u_X, l_X) ，右下角的格子為 (d_X, r_X) ， X 包含所有滿足 $u_X \leq i \leq d_X$ 且 $l_X \leq j \leq r_X$ 的格子 (i, j) 。

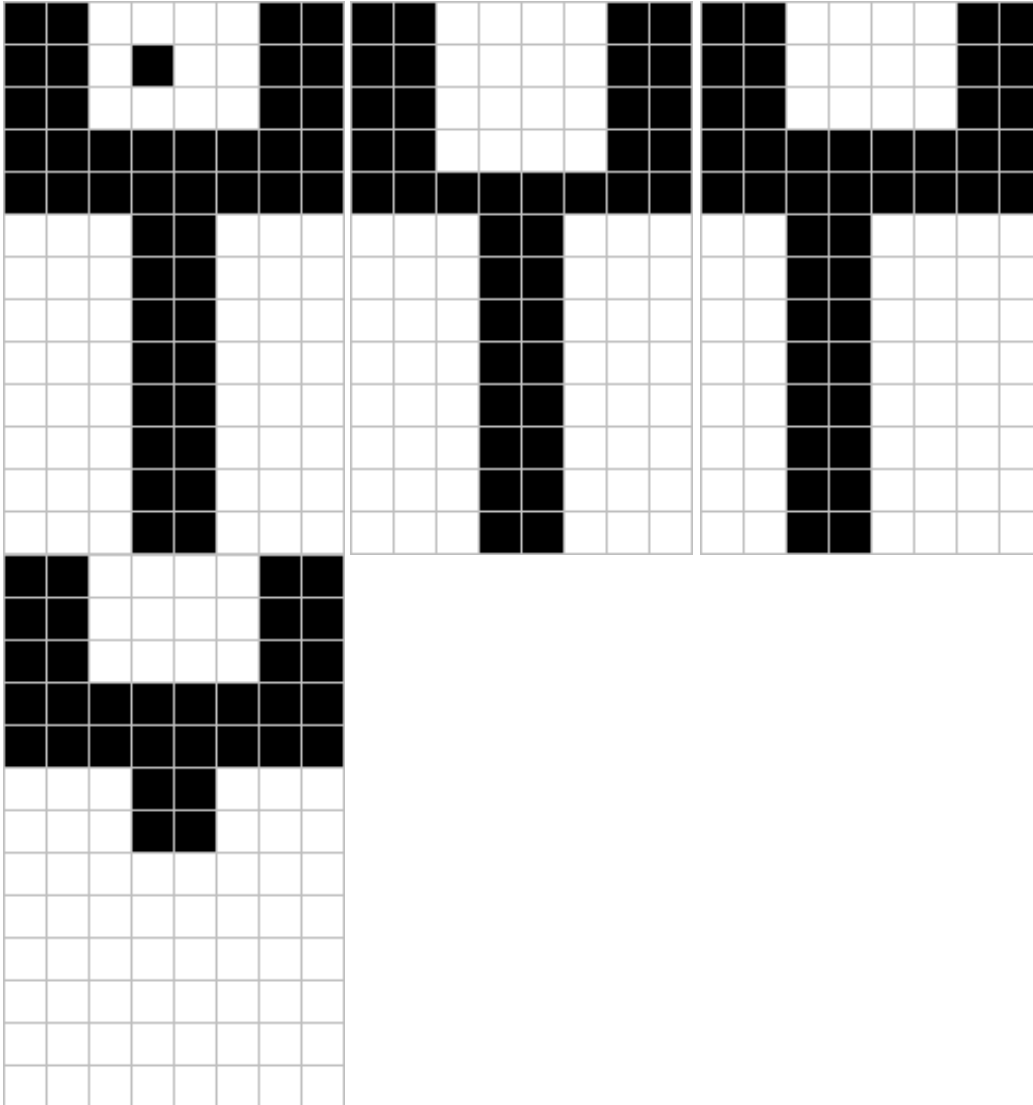
一個字母 Y 被分為四個矩形區域 A, B, C, D ，一張海報必須存在滿足以下所有條件的矩形區域 A, B, C, D 才能被視為一張印有字母 Y 的海報。

- $d_A - u_A > r_A - l_A \geq 0$ 。
- $d_B = d_A$ 且 $d_B - u_B = r_A - l_A$ 且 $l_B = r_A + 1$ 且 $r_B \geq l_B$ 。
- $d_C = d_A$ 且 $u_C = u_A$ 且 $l_C = r_B + 1$ 且 $r_C - l_C = r_A - l_A$ 。
- $u_D = d_A + 1$ 且 $r_B - r_D = l_D - l_B$ 且 $r_D - l_D = r_A - l_A$ 且 $d_D - u_D > r_D - l_D$ 。
- 對於所有在 A, B, C, D 任意一個矩形中的格子，其顏色皆為黑色；而對於所有不在 A, B, C, D 中的格子，其顏色皆為白色。

舉例來說，以下是一些符合條件的印有字母 Y 的海報。其中我們將黑色的格子塗上不同的顏色以辨識矩形，其中紅色的為矩形 A ，綠色的為矩形 B ，藍色的為矩形 C ，紫色的為矩形 D 。



而以下則是一些不符合條件的海報。

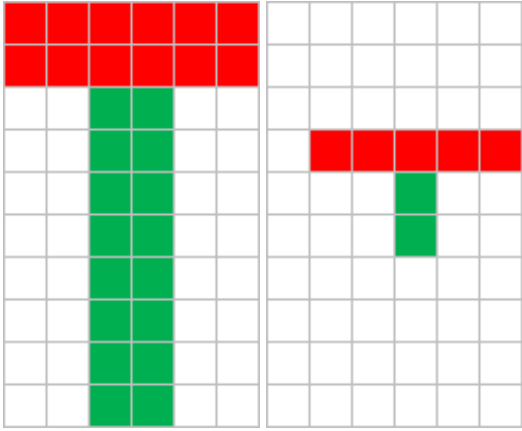


一個字母 **T** 被分為兩個矩形區域 E, F ，一張海報必須存在滿足以下所有條件的矩形區域 E, F 才能被視為一張印有字母 **T** 的海報。

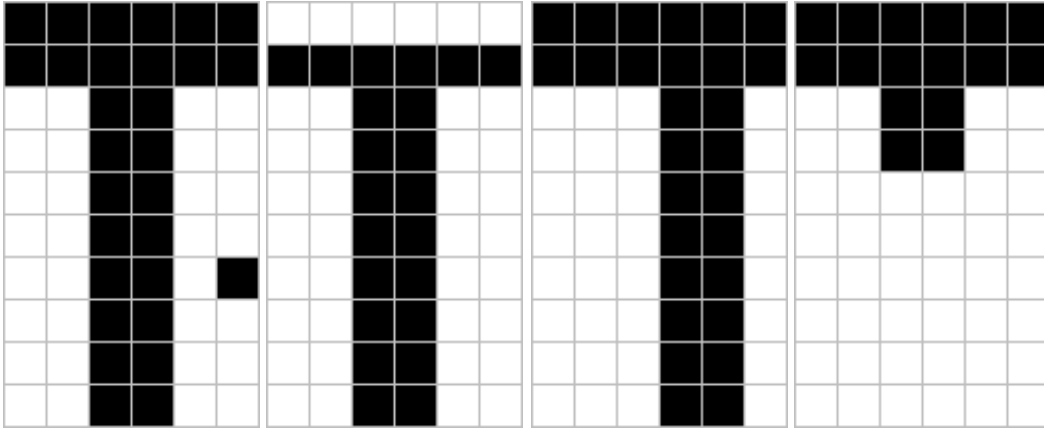
- $r_E - l_E > d_E - u_E \geq 0$ 。
- $u_F = d_E + 1$ 且 $r_E - r_F = l_F - l_E$ 且 $r_F - l_F = d_E - u_E$ 且 $d_F - u_F > r_F - l_F$ 。

- 對於所有在 E, F 任意一個矩形中的格子，其顏色皆為黑色；而對於所有不在 E, F 中的格子，其顏色皆為白色。

舉例來說，以下是一些符合條件的印有字母 T 的海報。其中我們將黑色的格子塗上不同的顏色以辨識矩形，其中紅色的為矩形 E ，綠色的為矩形 F 。



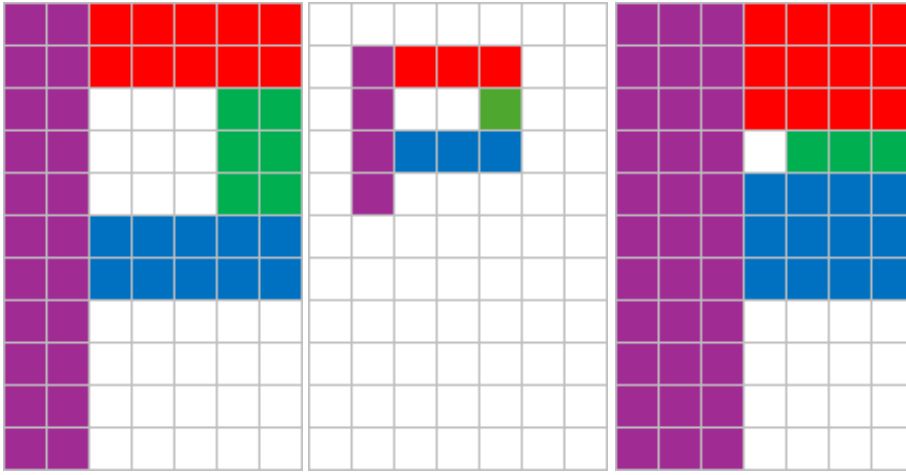
而以下則是一些不符合條件的海報。



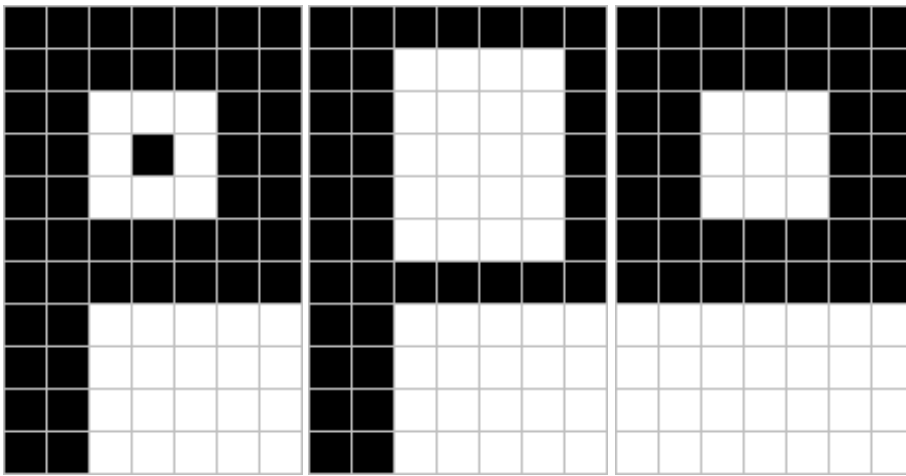
一個字母 P 被分為四個矩形區域 G, H, I, J ，一張海報必須存在滿足以下所有條件的矩形區域 G, H, I, J 才能被視為一張印有字母 P 的海報。

- $r_G - l_G > d_G - u_G \geq 0$ 。
- $u_H = d_G + 1$ 且 $d_H \geq u_H$ 且 $r_H = r_G$ 且 $r_H - l_H = d_G - u_G$ 。
- $u_I = d_H + 1$ 且 $d_I - u_I = d_G - u_G$ 且 $r_I = r_G$ 且 $l_I = l_G$ 。
- $u_J = u_G$ 且 $d_J > d_I$ 且 $r_J = l_G - 1$ 且 $r_J - l_J = d_G - u_G$ 。
- 對於所有在 G, H, I, J 任意一個矩形中的格子，其顏色皆為黑色；而對於所有不在 G, H, I, J 中的格子，其顏色皆為白色。

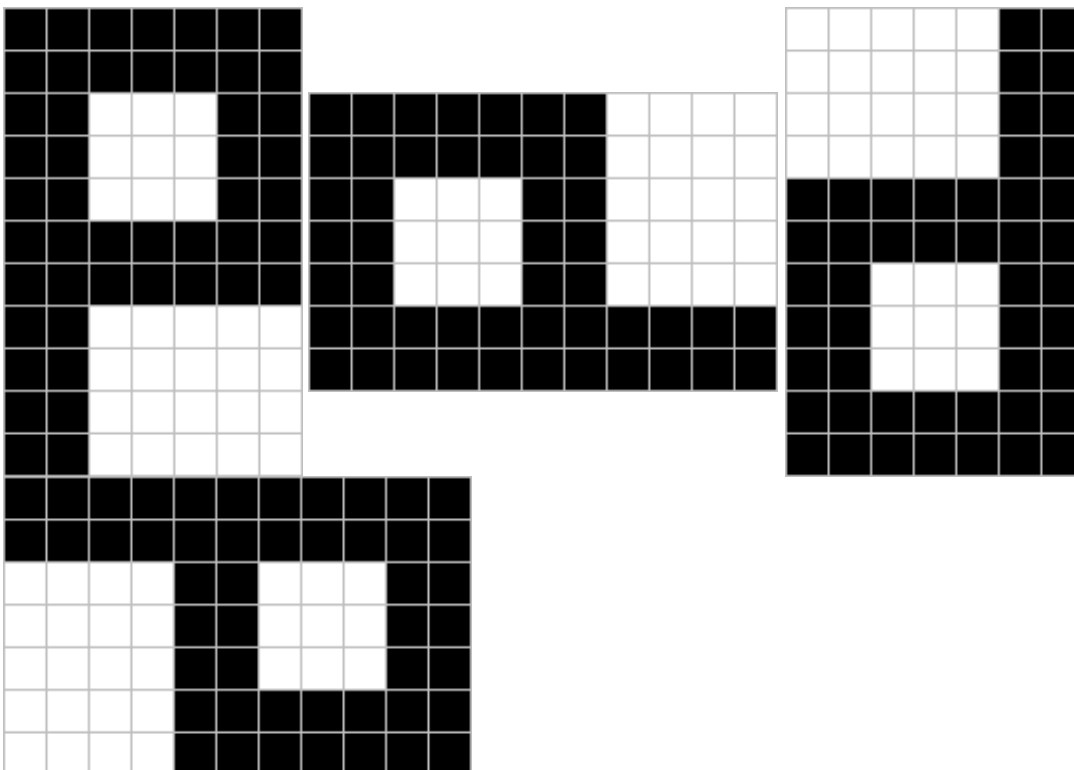
舉例來說，以下是一些符合條件的印有字母 P 的海報。其中我們將黑色的格子塗上不同的顏色以辨識矩形，其中紅色的為矩形 G ，綠色的為矩形 H ，藍色的為矩形 I ，紫色的為矩形 J 。



而以下則是一些不符合條件的海報。



由於海報被風吹散到各處，撿回的海報可能被旋轉了 0、90、180、或 270 度。每一張海報你都必須要檢查四個角度，只要其中一個旋轉的角度有符合字母的條件，就會被視為印有該字母。舉例來說，以下四張圖分別是被旋轉了 0、90、180、270 度的印有字母 P 的海報。



現在是時候寫一支程式來辨識海報上的字母了！請幫幫小 Y、小 T、小 P，好讓 YTP 活動能順利進行吧！

輸入格式

輸入的第一行包含兩個正整數 N, M ，以一個空白隔開，表示海報的大小。

接下來 N 行，每行包含一個長度為 M 的字串。其中第 i 行的第 j 個字元表示格子 (i, j) 的塗色狀況，若該字元為 # 表示該格為黑色，若該字元為 . 則表示該格為白色。

輸出格式

輸出一個字母，若該海報上印有 Y、T、P 字母，請輸出其對應的大寫字母；若該張海報無法被辨認出字母的海報，則請輸出大寫字母 X。

資料範圍

- $1 \leq N, M \leq 50$
- 字串僅由 # 及 . 組成

子任務

- 子任務 1 滿足輸入中只會有包含字母 T 或 P 的海報，且所有的海報都可以被辨識出字母。
- 子任務 2 滿足輸入中所有的海報都可以被辨識出字母。
- 子任務 3 無額外限制。

測試範例

輸入範例 1

```
13 8
##....##
##....##
##....##
#####
#####
...##...
...##...
...##...
...##...
...##...
...##...
...##...
...##...
```

輸出範例 1

```
Y
```

輸入範例 2

```
10 6
.....
.....
.....
.#####
...#..
...#..
.....
.....
.....
.....
```

輸出範例 2

```
T
```

輸入範例 3

```
7 11
#####....
#####....
##...##....
##...##....
##...##....
#####
#####
```

輸出範例 3

```
P
```

輸入範例 4

```
11 7
##.....
##.....
##.....
##.....
#####
#####
##...##
##...##
##...##
#####
#####
```

輸出範例 4

X

輸入範例 5

```
14 26
.....
.....##.....##...
.##....##.#####.##.###.
.##....##.#####.##.###.
.##....##.##.....##...##.
.##....##.##...##.##...##.
.###...###.##...###.##...###.
.#####.#####.#####.
...###.##...#####.#####.
.....##.....##.....
.....##.....##.....
...####.....##.....
...###.....##.....
.....
```

輸出範例 5

X

範例說明

- 範例輸入 1 為題目中第一張字母 Y 的海報。
- 範例輸入 2 為題目中第二張字母 T 的海報。
- 範例輸入 3 為題目中字母 P 旋轉 90 度的海報。
- 範例輸入 4 和 5 無法被辨識出字母，故輸出 X。

11_水果優勢(Fruit_Advantage)

(6 分 / 19 分)

時間限制: 2 seconds

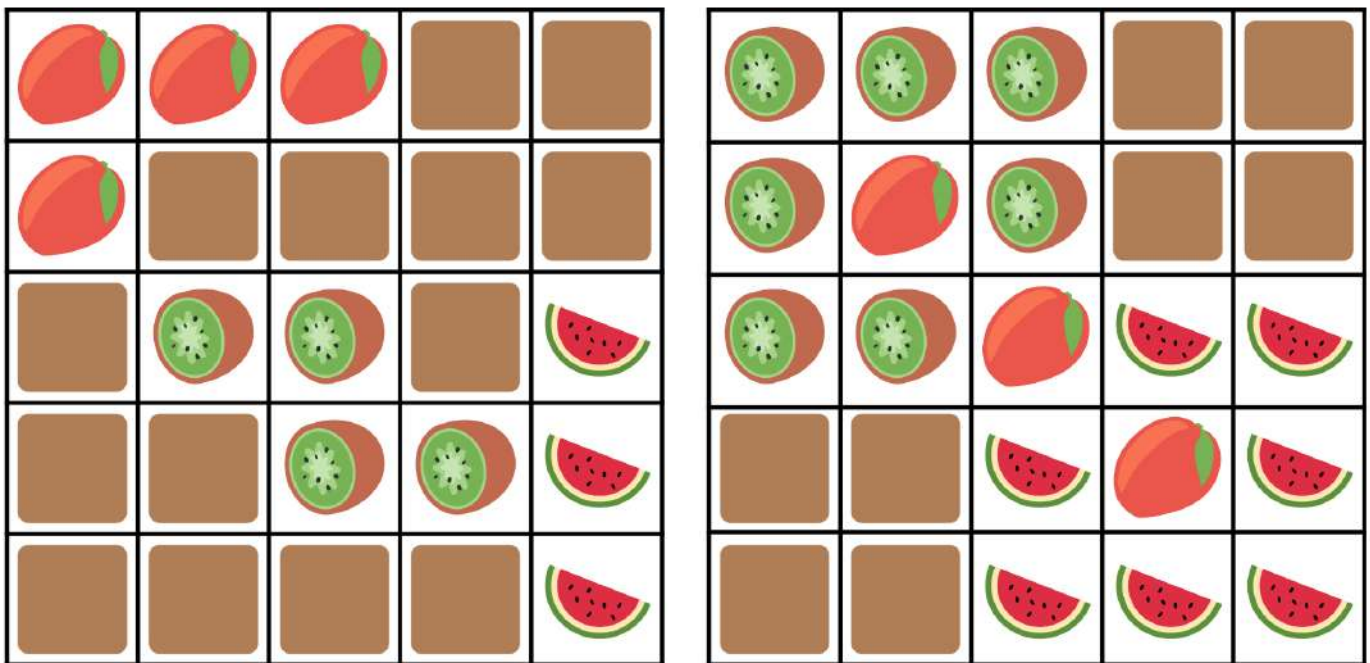
記憶體限制: 256 MB

題目敘述

巫醫巫醫有塊田，田可以用一個 $N \times M$ 表格來表示，每一個格子都可以種一種水果。不久前，巫醫巫醫在這塊田種了 K 種水果，為了方便她照顧，每一種水果種植的地方都會是一個連通塊。定義兩個格子相鄰若它們有一條共用邊，我們稱第 i 種水果是一個連通塊。若對於任意兩個種植第 i 種水果的格子 s, t ，存在一個由種植第 i 種水果的格子組成的序列 $c_1, c_2, \dots, c_k$ 滿足：

- $c_1 = s$
- $c_k = t$
- 對所有 $1 \leq i < k$ ， c_i 與 c_{i+1} 相鄰

舉例來說，下圖咖啡色的地方皆為空的田地。則左圖為一個合法的種植方式，而右圖則不是，因為芒果不是一個連通塊。



現在到了水果成熟的時間，為了能夠讓其他人體會到每一種水果的優勢，巫醫巫醫決定要將每一種水果都採收至少一顆。巫醫巫醫會使用機器人採收水果，每個機器人都有兩種採收模式，可以選擇其中一項執行：

- 選擇一個直行，並將該直行上的水果全部採收
- 選擇一個橫列，並將該橫列上的水果全部採收

因為機器人售價不菲，巫醫巫醫希望用最少的機器人達成她的目標，你能幫幫她嗎？

輸入格式

輸入第一行有三個正整數 N, M, K ，代表表格的大小以及水果的種類。

接下來 N 行，第 i 行有 M 個整數 $a_{i,1}, a_{i,2}, \dots, a_{i,M}$ ，其中 $a_{i,j}$ 為種在第 i 個橫排的第 j 個格子的水果種類。
若 $a_{i,j} = 0$ 則代表該格子沒有種水果，否則代表該格子種著第 $a_{i,j}$ 種水果。

輸出格式

請輸出一個整數，代表巫醫巫醫最少要用幾個機器人。

資料範圍

- $1 \leq NM \leq 300$
- $1 \leq K \leq NM$
- $0 \leq a_{i,j} \leq K$
- 對於所有 $1 \leq x \leq K$ ，至少有一個 $a_{i,j} = x$ 。

子任務

- 子任務 1 滿足 $1 \leq N, M \leq 7$ (6 分)
- 子任務 2 沒有額外限制 (19 分)

測試範例

輸入範例 1

```
5 5 3
1 1 1 0 0
1 0 0 0 0
0 2 2 0 3
0 0 2 2 3
0 0 0 0 3
```

輸出範例 1

```
2
```

輸入範例 2

```
3 4 4
0 1 0 4
2 1 3 4
2 0 3 0
```

輸出範例 2

```
1
```

輸入範例 3

```
3 3 5
1 2 3
4 0 0
5 0 0
```

輸出範例 3

```
2
```

輸入範例 4

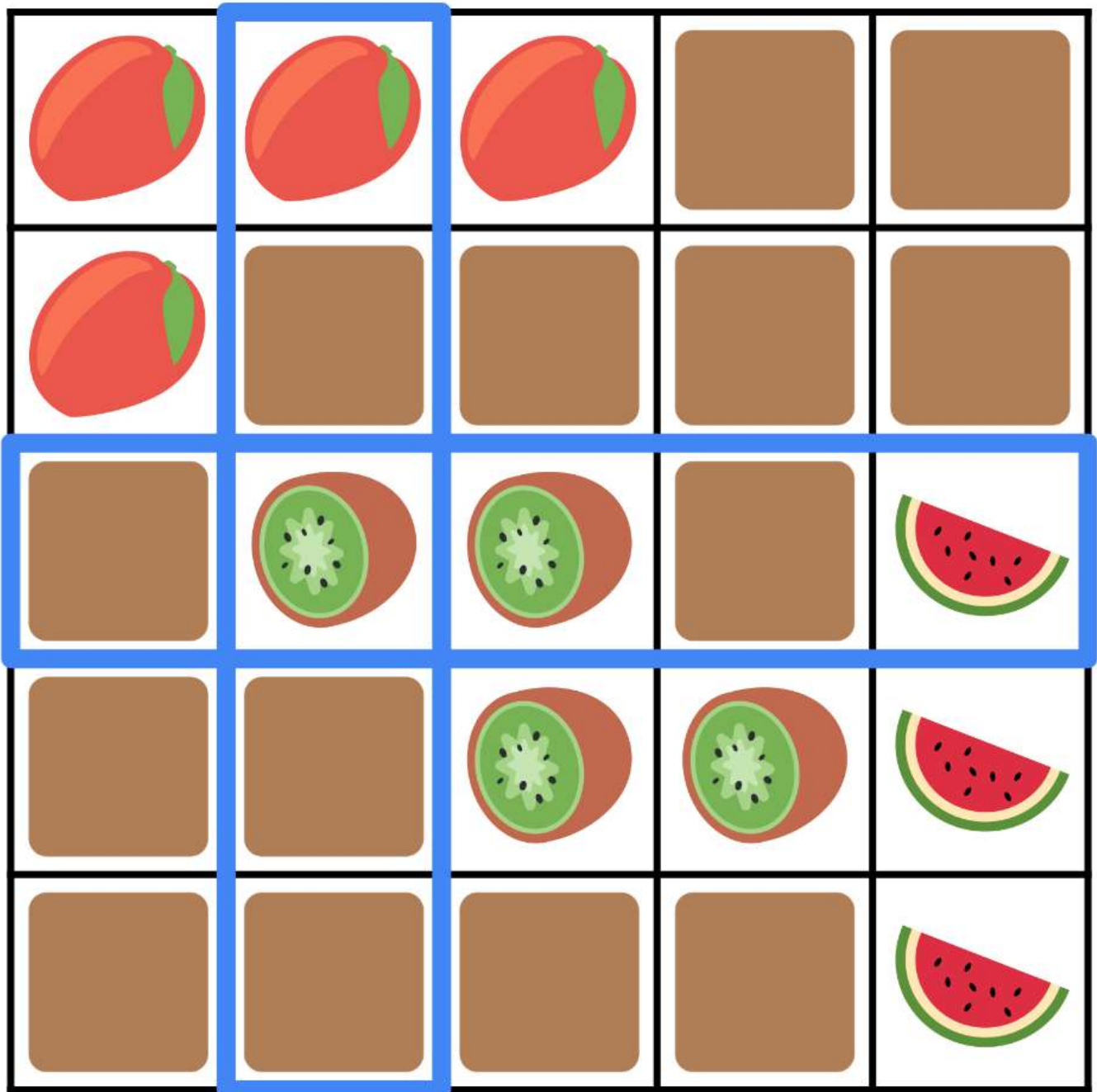
```
4 5 7
0 6 7 3 5
0 0 0 3 5
4 0 0 3 0
1 0 0 2 0
```

輸出範例 4

```
3
```

範例說明

在第一筆範例測資中，我們可以用兩個機器人採收藍色框框內的水果來達成目標，可以證明這個是其中一組最優解。



12_樓梯 (Staircase)

(3 分/8 分/7 分/7 分)

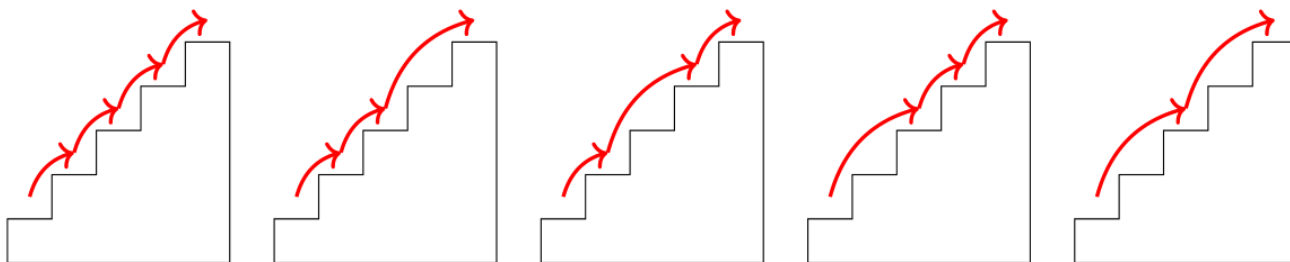
時間限制: 5 seconds

記憶體限制: 1024 MB

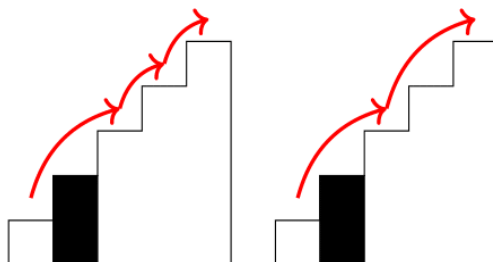
題目敘述

「從前從前有一座樓梯，每跨一步你可以往上走一階或兩階，請問從第一階走到第 x 階有幾種走法？」

如果兩種走法走過的樓梯階數的集合不一樣，則我們說兩種走法不一樣。聰明的你隨手畫出從第一階到第五階的每一種走法，很快就找到了規律。



你決定實際去嘗試走樓梯，印證你心中的答案。然而，樓梯被你來回踩了幾次，居然就這樣壞了！壞掉的那幾階樓梯在修好之前不能再踩上去，走樓梯的方法數也因此跟著變少了，例如，如果第二階壞了，那走到第五階就只剩兩種方法了。



樓梯斷了又修、修了又斷。現在給你第幾階樓梯斷掉或修好的紀錄，你能一邊算一算從第一階出發到某一階有幾種走法嗎？

輸入格式

輸入的第一行有兩個整數 N 、 Q ，分別代表樓梯最多有幾階、以及接下來的事件數量。

接下來的 Q 行每行代表一次事件：

- 1 x ：如果第 x 階現在是壞的，代表第 x 階修好了；否則，代表第 x 階壞掉了
- 2 x ：詢問若每一步可以往上走一或二階，從第一階走到第 x 階有幾種走法，途中不可以踩過壞掉且還沒修好的樓梯

輸出格式

對每次的操作二，輸出一個整數代表這次詢問的答案，對 998244353 取模。

資料範圍

- $N \leq 10^{18}$
- $Q \leq 5 \times 10^5$
- $1 \leq x \leq N$

子任務

- 子任務 1 滿足 $N, Q \leq 5000$ 。
- 子任務 2 只有第二種詢問，樓梯不會壞掉。
- 子任務 3 滿足同一時間至多只有十階是壞掉的還沒修好。
- 子任務 4 無額外限制。

測試範例

輸入範例 1

```
10 6
2 10
1 3
1 8
2 10
1 3
2 10
```

輸出範例 1

```
55
3
13
```

輸入範例 2

```
15 8
2 15
1 1
2 15
1 1
1 5
2 15
1 6
2 15
```

輸出範例 2

```
610
0
165
0
```

輸入範例 3

```
10 10
2 1
2 2
2 3
2 4
2 5
2 6
2 7
2 8
2 9
2 10
```

輸出範例 3

```
1
1
2
3
5
8
13
21
34
55
```

輸入範例 4

```
20 20
2 5
2 18
2 12
2 2
2 7
1 11
1 11
1 2
2 9
1 2
1 6
1 6
1 9
```

```
1 9
2 12
2 9
1 7
1 4
1 10
2 18
```

輸出範例 4

```
5
2584
144
1
13
13
144
34
42
```

範例說明

對於範例輸入二，如果第一階壞掉，沒有任何地方可以作為第一步的落腳點，因此走法數是零。

13_邏輯電路(Logic_Circuit)

(7分/9分/9分)

時間限制: 1 second

記憶體限制: 256 MB

題目敘述

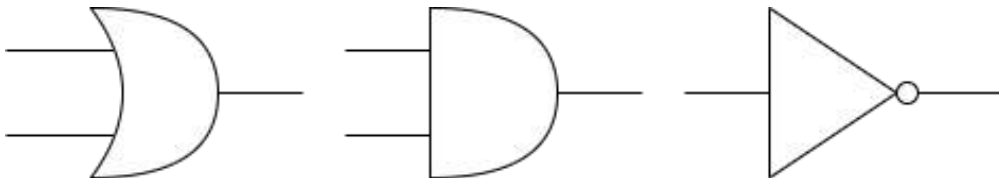
數位系統與實驗是許多大學資工系都會開設的課程，其中布林代數及數位電路的繪製是最基本的內容之一。在這道題目中，我們將完成一支程式，將輸入的邏輯運算式轉換成邏輯電路圖。還不會布林代數嗎？不用擔心，接下來我們進行簡單的教學。

在本題中，我們統一將邏輯運算式的輸出以大寫字母 **F** 作為變數名稱，並以剩下的 25 個大寫字母作為其他輸入的變數名稱。就如同程式語言中的邏輯運算，布林代數中最主要的三個運算子為 **OR**、**AND**、**NOT**，接下來我們將依序介紹它們。

在布林代數中，我們以類似數學中加法的表現方式表示 **OR** 運算。舉例來說，若將變數 **A** 和 **B** 進行 **OR** 運算，我們會寫作 **A+B**。而 **AND** 運算我們則以類似數學中乘法的表現方式表示，但我們會省略掉乘號，直接將兩個變數並列。舉例來說，若將變數 **A** 和 **B** 進行 **AND** 運算，我們會寫作 **AB**。最後則是 **NOT** 運算，我們在要取反的變數或運算式後加上一個單引號表示。舉例來說，若將變數 **A** 進行 **NOT** 運算，我們會寫作 **A'**。

而這三個運算子的運算優先順序也與大多數程式語言相同，首先優先計算括號內容，接下來先進行 **NOT** 運算，再進行 **AND** 運算，最後進行 **OR** 運算。

接下來將介紹如何將一個邏輯運算式轉換為邏輯電路圖。首先，我們先介紹三種運算子對應的邏輯閘。下圖由左至右分別為 **OR**、**AND**、**NOT** 的邏輯閘圖示，左側的線段表示邏輯閘的輸入，右側的線段表示邏輯閘的輸出。其中 **OR** 和 **AND** 的輸入可能有兩個或以上，而 **NOT** 只會有一個輸入；所有邏輯閘都只有一個輸出。



為了精確地定義輸入格式，我們保證在本題中，所有的輸入皆遵守以下巴科斯範式（Backus Normal Form）的 **<input>** 非終端符（Non-Terminal Symbol）：

```
<input> ::= "F=" <disj>
<disj>  ::= <conj> | <disj> "+" <conj>
<conj>  ::= <atom> | <conj> <atom>
<atom>  ::= <var> | <var> "'" | "(" <disj> ")" | "(" <disj> ")"'
<var>   ::= "A" | "B" | "C" | "D" | "E" | "G" | "H" | "I" | ... | "Z"
```

巴科斯範式的說明如下：

- **<input>**：表示本題的輸入字串。其中 **F** 為邏輯運算式的輸出，而 **<disj>** 則是主要的邏輯運算式內容。
- **<disj>**：由一連串的 **<conj>** 組成，**<conj>** 之間透過 **+** 字元隔開。表示邏輯運算式之間進行 **OR** 運算。
- **<conj>**：由一連串的 **<atom>** 組成，**<atom>** 之間沒有任何字元分隔。表示邏輯運算式之間進行 **AND** 運算。

- **<atom>**：可以是一個 **<var>** 或一個加了左右括號的 **<disj>**，並且可能加上了單引號表示對其進行 NOT 運算。
- **<var>**：表示邏輯運算式的輸入的變數名稱。注意 **F** 不在此列表內。

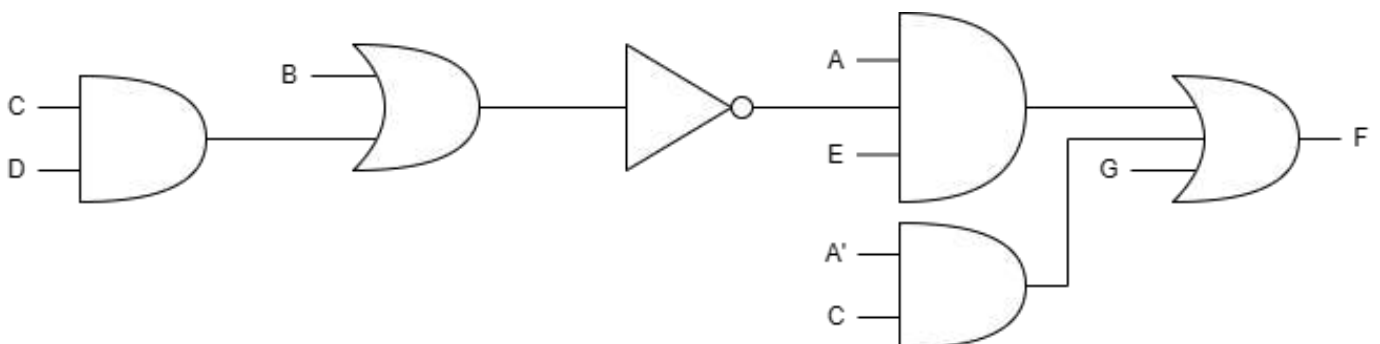
而以下是每個非終端符繪製成邏輯電路圖的方式：

- **<input>** 中的 **<disj>** 為主要的電路，其輸出為變數 **F**。
- 對於每個 **<disj>**，我們將其分解成一連串的 **<conj>**。
 - 如果 **<disj>** 只由一個 **<conj>** 組成，則將其視為一個 **<conj>**。
 - 否則，將這些 **<conj>** 的輸出，由上到下依序接到一個 OR 邏輯閘作為其輸入。
- 對於每個 **<conj>**，我們將其分解成一連串的 **<atom>**。
 - 如果 **<conj>** 只由一個 **<atom>** 組成，則將其視為一個 **<atom>**。
 - 否則，將這些 **<atom>** 的輸出，由上到下依序接到一個 AND 邏輯閘作為其輸入。
- **<atom>** 則依照不同情況進行繪製。
 - **<var>**：為一變數，寫下該變數的名稱。
 - **<var> " ' "**：為一變數取反，寫下該變數的名稱，並加上單引號。
 - **" (" <disj> ") "**：為一個新的子運算式，遞迴繪製此運算式的內容。
 - **" (" <disj> ") ' "**：為一個新的子運算式並取反，遞迴繪製此運算式的內容，並將其輸出接上一個 NOT 邏輯閘。

舉例來說，輸入 $F = A(B+CD)'E + A'C + G$ ，則繪製邏輯電路圖的過程如下：首先得到輸入的主要電路為 $A(B+CD)'E + A'C + G$ 。將其分解成三個 **<conj>**： $A(B+CD)'E$ 、 $A'C$ 、 G 。

- 將 $A'(B+CD)'E$ 分解成三個 **<atom>**： A' 、 $(B+CD)'$ 、 E 。
 - A' 為一變數取反，寫下變數名稱，並加上單引號。
 - $(B+CD)'$ 為運算式取反，遞迴繪製運算式 $B+CD$ ，並將其輸出接上一個 NOT 邏輯閘。
 - E 為一變數，寫下變數名稱。
- 將 $A'C$ 分解成兩個 **<atom>**： A' 、 C 。
 - A' 為一變數取反，寫下變數名稱，並加上單引號。
 - C 為一變數，寫下變數名稱。
- G 只由一個 **<atom>** 組成，將其視為一個 **<atom>**。此 **<atom>** 為一變數，寫下變數名稱。

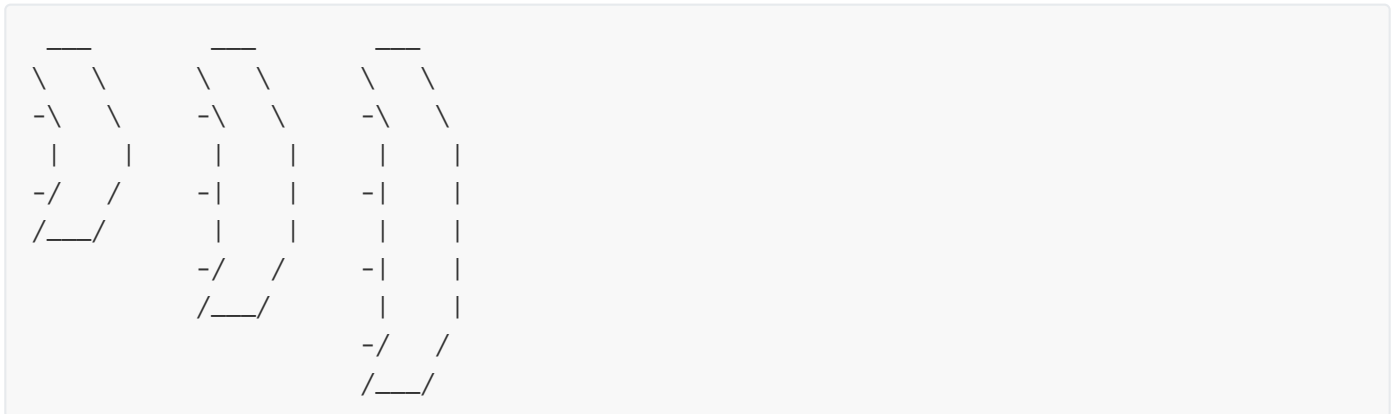
繪製出的電路圖如下圖：



學會了如何繪製之後，最後我們要來將邏輯電路圖利用 ASCII Art 的方式輸出到螢幕上！接下來我們會先說明如何繪製三種不同的邏輯閘，最後再說明將其接成電路圖時，邏輯閘之間的排列方式。

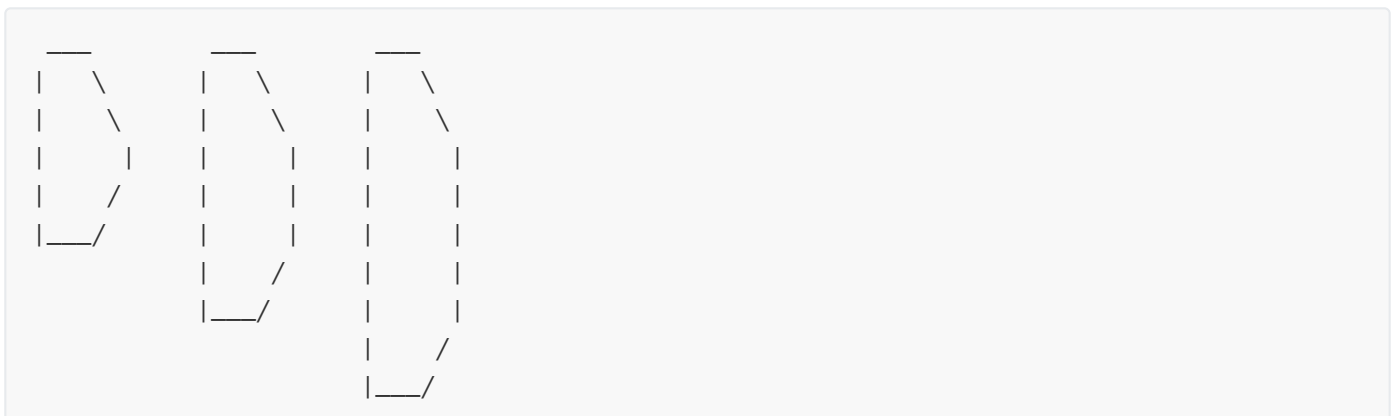
首先是 OR 邏輯閘。對於一個有 n 個輸入的 OR 邏輯閘，其會佔用一個 $2n + 2$ 行高，7 個字元寬的矩形範圍。邏輯閘第 1 行的第 2, 3, 4 個字元為底線 (_) ；第 2 行的第 1, 5 個字元為反斜線 (\) ；第 3 行的第 1 個字元為減號 (-) ，第 2, 6 個字元為反斜線 (\) ；第 4 到第 $2n$ 行的第 2, 7 個字元為豎線 (|) ，並且對於這之間的所有奇數行，第 1 個字元為減號 (-) ；第 $2n + 1$ 行的第 1 個字元為減號 (-) ，第 2, 6 個字元為斜線 (/) ；第 $2n + 2$ 行的第 1, 5 個字元為斜線 (/) ，第 2, 3, 4 個字元為底線 (_) 。

舉例來說，下方由左到右分別是有 2, 3, 4 個輸入的 OR 邏輯閘。邏輯閘的輸入位於第 3 行開始的每個奇數行的左側，而輸出則位於第 $n + 2$ 行的右側。請注意，左側的減號也視為邏輯閘的一部分。



接下來是 AND 邏輯閘。對於一個有 n 個輸入的 AND 邏輯閘，其會佔用一個 $2n + 2$ 行高，7 個字元寬的矩形範圍。邏輯閘第 1 行的第 2, 3, 4 個字元為底線 ()；第 2 行的第 1 個字元為豎線 (|)，第 5 個字元為反斜線 (\)；第 3 行的第 1 個字元為豎線 (|)，第 6 個字元為反斜線 (\)；第 4 到第 $2n$ 行的第 1, 7 個字元為豎線 (|)；第 $2n + 1$ 行的第 1 個字元為豎線 (|)，第 6 個字元為斜線 (/)；第 $2n + 2$ 行的第 1 個字元為豎線 (|)，第 2, 3, 4 個字元為底線 ()，第 5 個字元為斜線 (/)。

舉例來說，下方由左到右分別是有 2, 3, 4 個輸入的 AND 邏輯閘。邏輯閘的輸入位於第 3 行開始的每個奇數行的左側，而輸出則位於第 $n + 2$ 行的右側。



接下來是 NOT 邏輯閘。一個 NOT 邏輯閘會佔用一個 6 行高，7 個字元寬的矩形範圍。邏輯閘第 1 行的第 2, 3 個字元為底線 (_)；第 2 行的第 1 個字元為豎線 (|)，第 4 個字元為反斜線 (\)；第 3 行的第 1 個字元為豎線 (|)，第 5 個字元為反斜線 (\)；第 4 行的第 1 個字元為豎線 (|)，第 6 個字元為右括號 ())，第 7 個字元為小寫字母 o；第 5 行的第 1 個字元為豎線 (|)，第 5 個字元為斜線 (/)；第 6 行的第 1 個字元為豎線 (|)，第 2, 3 個字元為底線 (_)，第 4 個字元為斜線 (/)。

下方為一個 NOT 邏輯閘。邏輯閘的輸入位於第 4 行的左側，而輸出則位於第 4 行的右側。



接下來我們要說明如何輸出一個變數。每個變數或變數取反的輸出會佔用一個 1 行高，4 個字元寬的矩形範圍。第 1 個字元為變數名稱，第 2 個字元若為變數取反則為單引號 (')，否則留空，第 3, 4 個字元為減號 (-)。

舉例來說，下方分別是變數 A 及其取反，其輸出位於右側。

A -- A' --

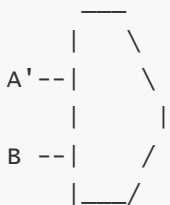
最後，我們要來說明如何將邏輯閘組裝成完整的邏輯電路。我們會用遞迴的方式進行繪製邏輯電路：先繪製邏輯閘，再由上往下依序繪製每一個輸入接到的邏輯電路。為了方便進行電路繪製時是否有重疊區域的判斷，我們定義一個邏輯電路佔據的矩形區域為，最小可以覆蓋住所有邏輯閘及變數和變數取反的矩形區域。

對於一個有 n 個輸入的邏輯閘由上往下的第 i 個輸入，若該輸入為一變數或變數取反，則直接繪製變數或變數取反的矩形於邏輯閘左側，與輸入位置相同的高度。若該輸入為另一個電路，則其矩形應於邏輯閘的左側，兩者左右之間留下 $2n + 3$ 個空格作為稍後進行連接使用。接下來決定電路的上下位置，首先將電路上下移動，使其輸出位置的高度與邏輯閘第 i 個輸入的高度相同，再調整高度，若其與其他已繪製的電路所佔的矩形有重疊部分，則向下移動直到沒有重疊為止。最後進行邏輯閘間的連接，如果電路的輸出與邏輯閘的輸入高度相同，則在兩者左右之間的空格補上 $2n + 3$ 個減號 (-)；如果兩者高度不同，則先在電路輸出的右側空格由左至右依序補上 $2i$ 個減號 (-) 及 1 個加號 (+)，並在邏輯閘輸入的左側由右至左依序補上 $2(n - i) + 2$ 個減號 (-) 及 1 個加號 (+)，此時兩個加號應位於同一欄，最後在兩加號上下之間的空格補滿豎線 (|)。

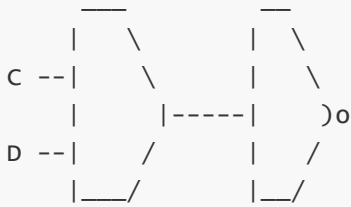
繪製完完整的邏輯電路後，在電路的輸出位置依序加上 2 個減號 (-) 及 1 個空格，最後寫上變數 F，表示整個電路的輸出。

舉例來說，繪製 $F = A'B + (CD)'$ 的邏輯電路圖時，我們先繪製一個 OR 邏輯閘，接下來依序繪製 $A'B$ 和 $(CD)'$ 。

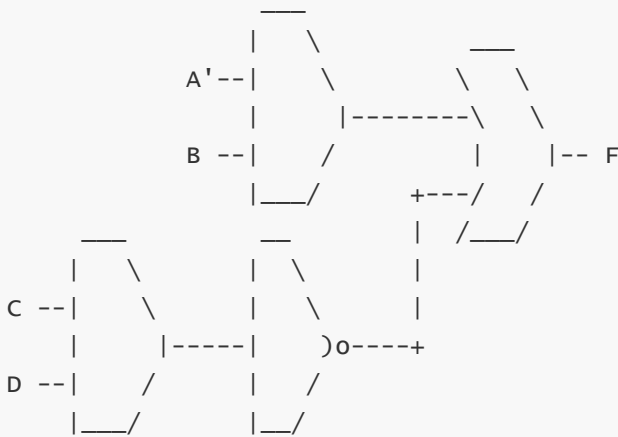
繪製 $A'B$ 時，先繪製一個 AND 邏輯閘，並且由於 A' 及 B 分別為變數取反及變數，我們直接將其接上 AND 邏輯閘，完成 $A'B$ 的繪製。 $A'B$ 電路圖繪製如下：



而 $(CD)'$ 我們會先繪製一個 NOT 邏輯閘，再繪製 CD 部分。由於 NOT 邏輯閘只有 1 個輸入，其與 CD 部分之間間隔為 5 格，並且由於電路輸出及邏輯閘輸入高度相同，故用減號補上之間的空格。 $(CD)'$ 部分的電路與電路圖繪製如下：



最後，依序將兩者接上 OR 邏輯閘。 $A'B$ 與 OR 邏輯閘左右之間應間隔 7 格，由於電路輸出與邏輯閘輸入之間高度相同，故用減號補上之間的空格。而 $(CD)'$ 與 OR 邏輯閘左右之間應間隔 7 格，並且為了避免與上方的 $A'B$ 重疊， $(CD)'$ 會向下移動 3 行，並依上述格式連接至 OR 邏輯閘。最後再補上電路輸出 F 即大功告成。



接下來便輪到你寫一支程式進行轉換了，加油！

輸入格式

輸入僅包含一字串 S ，表示要轉換的邏輯運算式。

輸出格式

輸出的第一行包含兩個整數 H, W ，以一個空白隔開。依序表示邏輯電路圖的行數及每一行的字元數。

接下來請輸出 H 行，每行包含 W 個字元，為轉換後的邏輯電路圖。

請注意，你的輸出必須為邏輯電路圖佔據的最小矩形區域，並且所有的空白都必須輸出，包含行尾空白。此題將以嚴格比對的方式評測。

資料範圍

- $3 \leq |S| \leq 100$
- 保證 S 遵守題目敘述中的巴科斯範式

子任務

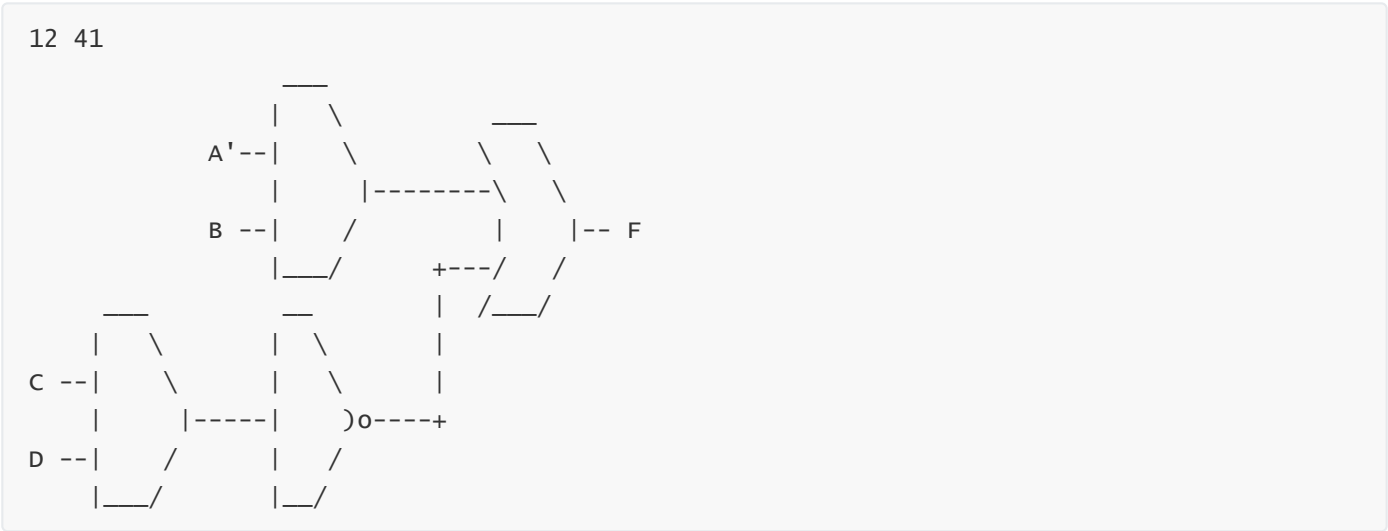
- 子任務 1 滿足 $|S| \leq 5$
- 子任務 2 滿足 S 中不會有任何單引號 (`'`)
- 子任務 3 無額外限制

測試範例

輸入範例 1

F=A'B+(CD)'

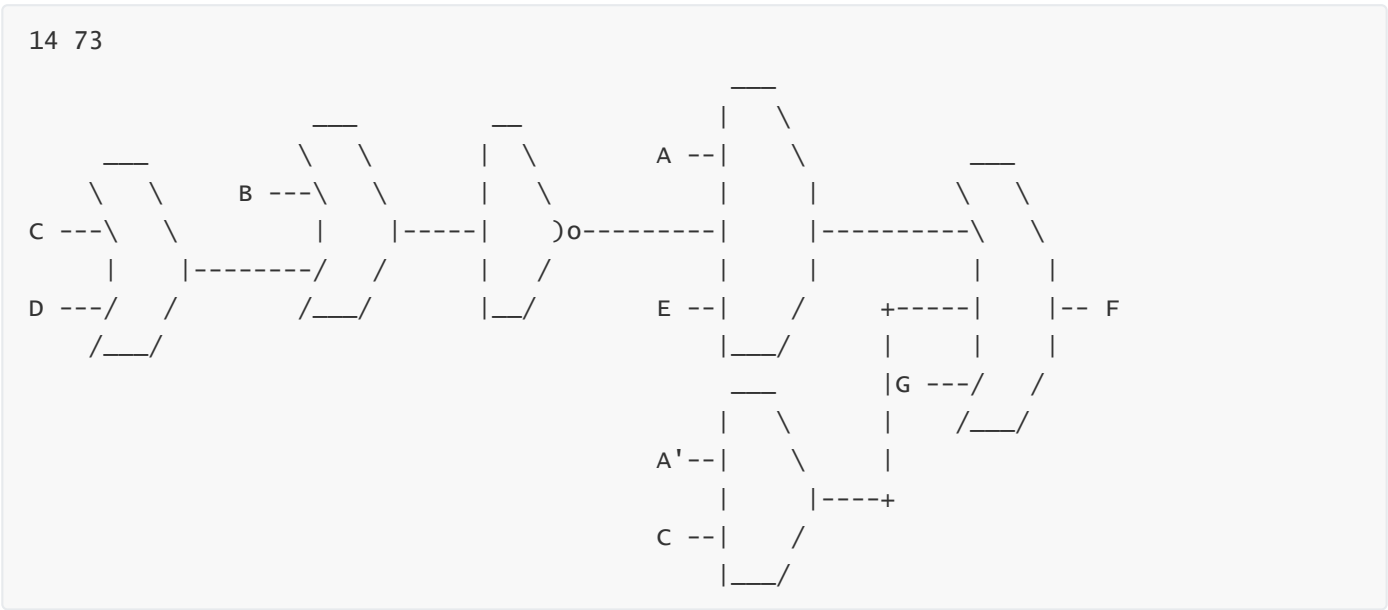
輸出範例 1



輸入範例 2

F=A(B+(C+D))'E+A'C+G

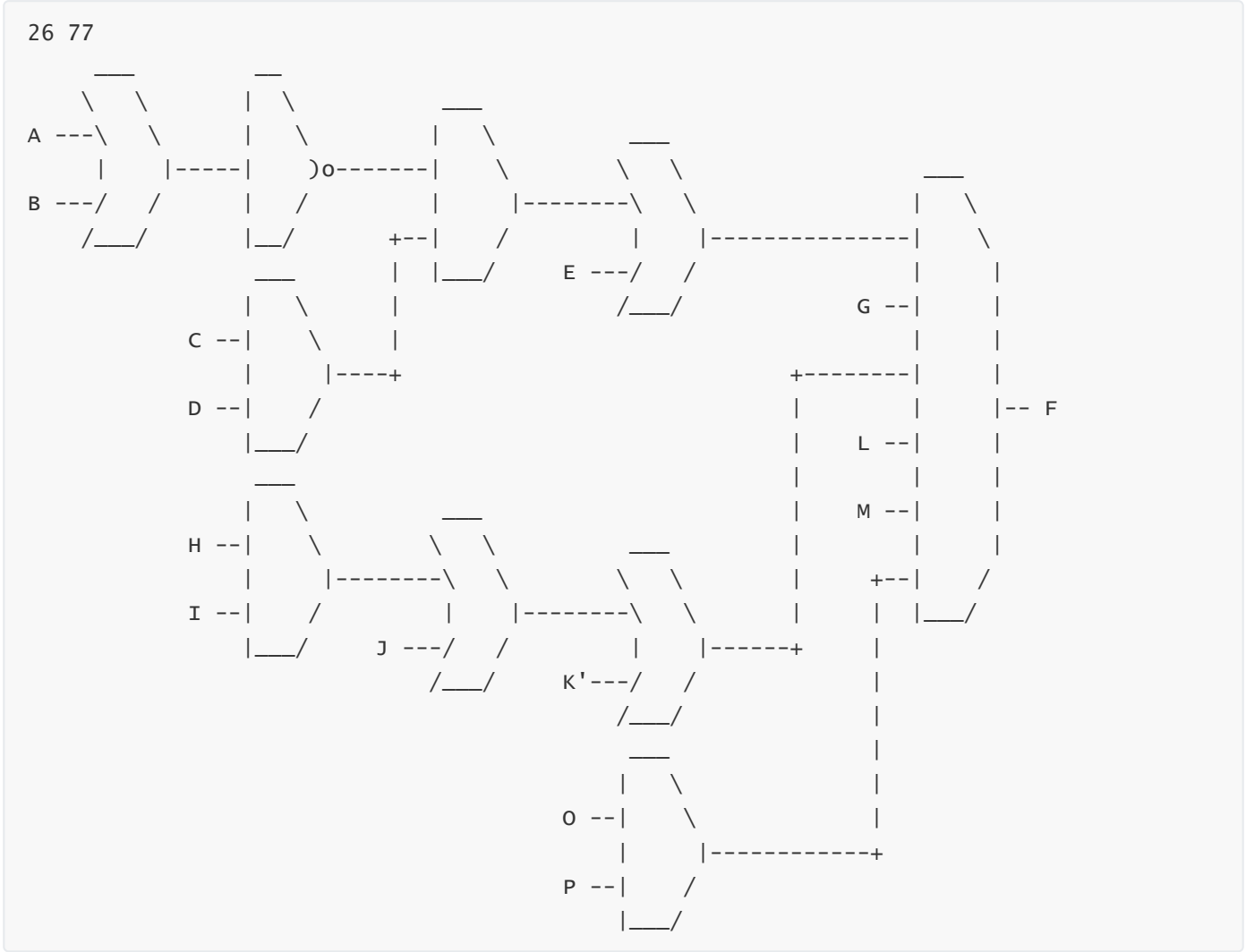
輸出範例 2



輸入範例 3

$$F = ((A+B)'(CD)+E)G((HI+J)+K')LM(OP)$$

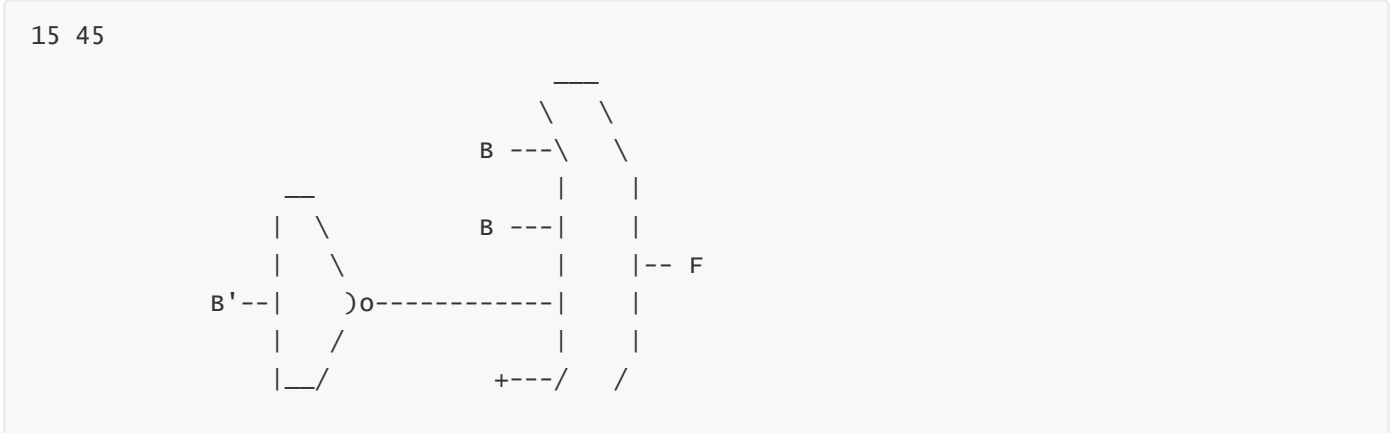
輸出範例 3



輸入範例 4

$$F = B + (B) + (B')' + ((B)')'$$

輸出範例 4





輸入範例 5

F=X

輸出範例 5

1 8
X ---- F

範例說明

輸出範例 3 中，注意 B' 及 $(B)'$ 根據巴科斯範式的定義，兩者繪製出的電路圖是不同的。

另外，本題無須簡化電路圖或運算式，請直接將給定的邏輯運算式依照題目敘述中的規則繪製出邏輯電路圖。