
# 送分題 – Hello World

***(30分)***

## 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！

## 問題描述

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, 以及及常用英文符號 。

## 資料範圍

[A~Z][a~z], 空格, 以及驚嘆號 “!”

## 資料範例

## 輸入範例1

(無輸入值)

## 輸出範例1

Hello World!

# 範例解釋

輸入範例1, 無輸入值，簡單而快樂的輸出Hello World!

# Freebit question – Hello World

***(30 points)***

## Introduction

YTP Contest has started!

Let’s verify everything first.

Is the internet setting correct?

Is the source code submission working well?

Do you use STDOUT output for program solutions?

If everything is ready, 30 points are yours! Go! Go! Go!

## Description

Please write a program to output Hello World!

## Input Format

This problem requires no input.

## Output Format

[A~Z][a~z], space, and common English punctuation.

## Constraints

[A~Z][a~z], space, and exclamation mark "!".

## Data Examples

## Input Example 1

(no input)

## Output Example 1

Hello World!

# Example Explanation

Input Example 1 has no input, simply output Hello World!

# 撲克之王安妮亞 - Poker King Anya

***(10分)***

## 問題敘述

小女孩安妮亞最近在學習撲克牌的規則，她不只是聰明，還學得很快速。因此他發明了一個簡單的遊戲：從撲克牌的牌庫堆抽 5 張牌，然後把這些牌對應到的點數加總。你可以幫安妮亞找到答案嗎？

## 輸入格式

輸入只有一行，有五個整數，分別代表安妮亞抽出的五張牌。每個整數之間以空白隔開。

## 輸出格式

輸出答案後換行，答案會是一個整數。

## 資料範圍

所有輸入的整數介於 1 到 13 （含）。

## 資料範例

### 輸入範例 1

```
1 1 1 1 2
```

### 輸出範例 1

```
6
```

### 輸入範例 2

```
10 3 1 9 13
```

## 輸出範例 2

```
36
```

## 輸入範例 3

```
13 13 12 12 11
```

## 輸出範例 3

```
61
```

# 範例說明

### 範例 1

1 + 1 + 1 + 1 + 2 = 6

### 範例 2

10 + 3 + 1 + 9 + 13 = 36

### 範例 3

13 + 13 + 12 + 12 + 11 = 61

# Poker King Anya

***(10 points)***

## Description

Little girl Anya is recently learning the rule of pokers. She is very smart and learning very fast. Thus she invented a simple game: draw 5 cards from the deck and then find the sum of the 5 cards' corresponding values. Can you help her find the answer?

## Input Format

There will be 5 integers in one line, indicating the corresponding values of Anya's 5 cards respectively. The integers are separated by spaces.

## Output Format

Output the answer and end with a newline character. The answer will be an integer.

## Constraints

All integers are between 1 and 13 (inclusive).

## Sample Test Cases

### Input Example 1

```
1 1 1 1 2
```

### Output Example 1

```
6
```

## Input Example 2

```
10 3 1 9 13
```

## Output Example 2

```
36
```

## Input Example 3

```
13 13 12 12 11
```

## Output Example 3

```
61
```

# Example Explanation

### Example 1

1 + 1 + 1 + 1 + 2 = 6

### Example 2

10 + 3 + 1 + 9 + 13 = 36

### Example 3

13 + 13 + 12 + 12 + 11 = 61

# 保齡球 - Bowling

***(10分)***

## 問題敘述

蛋餅是一位優秀的保齡球選手，普通的規則已經無法展現他的強大之處了，因此他發明了屬於自己的保齡球規則。在一般的保齡球規則中，如果全倒就會獲得 $10$ 分 + 下一球的得分 + 下下球的得分，蛋餅將它修改成「如果全倒，就會獲得『上一球的得分乘以 $c_1$』+『上上球的得分乘以 $c_2$』$+\ldots+$ $K$ 個球之前的得分乘以 $c_K$」（注意並沒有基本分 $10$ 分）。你趕到保齡球場的時候，蛋餅已經打完前 $K$ 球了，分數分別是 $a_1, a_2, \ldots, a_K$，並且他很有自信的覺得接下來的每一球都能打出全倒，因為你已經跟他約了吃拉麵，所以蛋餅只會打 $N$ 球。

另外，由於計分板在設計的時候並沒有想到有人這麼會打保齡球，因此他只會顯示蛋餅的總分除以 $10^9+9$ 的餘數。

幫蛋餅計算他打完 $N$ 球之後，計分板上顯示的分數吧。

## 輸入格式

輸入首行有兩個正整數 $N, K$，代表蛋餅總共要打的球數以及全倒可以獲得幾球之前的分數，同時也是蛋餅已經打完的球數。

接下來一行有 $K$ 個正整數 $a_1, \ldots, a_K$，代表蛋餅第 $i$ 球打出的分數為 $a_i$。

接下來一行有 $K$ 個正整數 $c_1, \ldots, c_K$，代表打出全倒時可以獲得前 $i$ 個球的分數乘以 $c_i$。

## 輸出格式

輸出一個非負整數於一行，代表蛋餅打完 $N$ 球之後，計分板顯示的分數。

## 資料範圍

- $1 \le K \le N \le 10^3$
- $1 \le a_i, c_i \le 10^9$

## 資料範例

## 輸入範例 1

```
5 3
9 8 3
9 2 5
```

## 輸出範例 1

```
946
```

## 輸入範例 2

```
3 1
5
3
```

## 輸出範例 2

```
65
```

## 輸入範例 3

```
2 2
1000000000 1000000000
1000000000 1000000000
```

## 輸出範例 3

```
999999991
```

# 範例說明

在第一筆範例中，蛋餅第 $4$ 球拿到 $3 \times 9 + 8 \times 2 + 9 \times 5 = 88$ 分，第 $5$ 球拿到 $88 \times 9 + 3 \times 2 + 8 \times 5 = 838$ 分，總分是 $9 + 8 + 3 + 88 + 838 = 946$ 分。

在第二筆範例中，蛋餅第 $2$ 球拿到 $5 \times 3 = 15$ 分，第 $3$ 球拿到 $15 \times 3 = 45$ 分，總分是 $5 + 15 + 45 = 65$ 分。

在第三筆範例中，總分 $10^9 + 10^9 = 2 \times 10^9$ 分，但是計分板只會顯示總分除以 $10^9 + 9$ 的餘數，因此會顯示 $999999991$ 分。

# Bowling

***(10 points)***

## Description

Omelet is an excellent bowling player so the ordinary rules can no longer show his strength. Therefore, he invented his own bowling rules. In the general bowling rules, you get "$10+$ the score of te next ball + the score of the next of  next ball" if you get a strike, and Omelet modified it to "the score of the previous ball mutiplied by $c_1$ + the score of the previous of the previous ball multiplied by $c_2$ + . . . + the score of the ball $K$ before multiplied by $c_K$"(note that there is no basic score $10$). When you arrive at the bowling alley, Omelet has played the first $K$ ball(s), and the score is $a_1, a_2, \ldots, a_K$. He is confident that he can get strike in all the next ball. Because Omelet is going to eat ramen with you, he will only play $N$ ball(s).

Also, since the scoreboard is not designed for someone who can bowl so well, it will only display the reminder of Omelet's total score divided by $10^9 + 9$.

Help Omelet compute the score displayed by the scoreboard after he plays $N$ ball(s).

## Input Format

The first line of the input contains two positive integers $N, K$, indicating the number of ball(s) Omelet is going to play and the number of balls which a strike can get score from. The latter is also the number of ball(s) Omelet has played.

The second line of the input contains $K$ positive integers $a_1, a_2, \ldots, a_K$, indicating the score of the $i$-th ball Omelet played.

The third line of the input contains $K$ positive integers $c_1, c_2, \ldots, c_K$, indicating the score of the $i$-th previous ball is multiplied by $c_i$ if Omelet gets a strike.

## Output Formet

Output a non-negative integer in one line, indicating that the remainder of the number of the score displayed by the scoreboard after Omelet play $N$ ball(s) divded by $10^9 + 9$.

## Constraints

- $1 \leq K \leq N \leq 10^3$
- $1 \leq a_i, c_i \leq 10^9$

# Sample Test Cases

## Input Example 1

```
5 3
9 8 3
9 2 5
```

## Output Example 1

```
946
```

## Input Example 2

```
3 1
5
3
```

## Output Example 2

```
65
```

## Input Example 3

```
2 2
1000000000 1000000000
1000000000 1000000000
```

## Output Example 3

```
999999991
```

# Example Explanation

In the first example, Omelet gets $3 \times 9 + 8 \times 2 + 9 \times 5 = 88$ points in the $4$th ball, and $88 \times 9 + 3 \times 2 + 8 \times 5 = 838$ points in the $5$th ball. The total score is $9 + 8 + 3 + 88 + 838 = 946$ points.

In the second example, Omelet gets $5 \times 3 = 15$ points in the $2$nd ball, and $15 \times 3 = 45$ points in the $3$rd ball. The total score is $5 + 15 + 45 = 65$ points.

In the third example, the total score is$10^9 + 10^9 = 2 \times 10^9$ points, but the scoreboard only displays the remainder of the total score divided by $10^9 + 9$, so the result is $999999991$ points.

# 星星樹 - Star Trees

***(10分)***

時間限制：1 秒

記憶體限制：512 MB

## 問題敘述

聖誕節就快要到了！小 T、小 O、小 J 跟小 E 一起，來到了傳說中人擠人的新北耶誕城。

一走出車站，除了擁擠的人潮，就是一排掛滿星星的聖誕樹了！每棵樹上都掛了許多星星，好奇的小 E 想知道，對於每一個 $i$，從左邊數來的前 $i$ 棵樹上，總共有多少星星。

當他好不容易算完以後，小 T 問他：那你知道，第 $l$ 棵樹到第 $r$ 棵樹中間，總共有多少星星嗎？

正當小 E 要重新開始數的時候，聰明的小 O 跳了出來，看著小 E 剛剛的計算過程，直接講出了答案！小 T 一連問了幾百次，小 O 都能馬上回答出來。

你知道，小 O 是怎麼做到的嗎？請先幫小 E 算出他的問題的答案，再試著模仿小 O，快速回答小 T 的問題吧！

## 輸入格式

輸入第一行有一個整數 $N$，代表總共有 $N$ 棵聖誕樹。

第二行有 $N$ 個整數 $a_i$ ，代表從左到右第 $i$ 棵樹上的星星數量。

第三行有一個整數 $Q$，代表小 T 問了幾個問題。

第 $4$ 到 $Q+3$ 行，每行有兩個整數 $l, r$，代表小 T 想要知道第 $l$ 棵樹到第 $r$ 棵樹中間(含 $l, r$)，共有幾顆星星。

## 輸出格式

第一行請輸出 $N$ 個整數，第 $i$ 個數代表從左邊數來的前 $i$ 棵樹上，總共有多少星星。

第 $2$ 到 $Q+1$ 行，每行請輸出一個整數，代表對小 T 問題的回答。

## 資料範圍

$1 \leq N, Q \leq 500000$

$1 \leq a_i \leq 100$

$1 \leq l \leq r \leq N$

# 資料範例

## 輸入範例 1

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

## 輸出範例 1

```
3 7 12 14
7
12
```

## 輸入範例 2

```
3
3 10 7
2
1 3
3 3
```

## 輸出範例 2

```
3 13 20
20
7
```

## 輸入範例 3

```
6
99 99 99 99 99 99
4
1 1
3 4
5 6
1 6
```

## 輸出範例 3

```
99 198 297 396 495 594
99
198
198
594
```

# 範例說明

範例輸入 1 中，從左邊數來的前 $1, 2, 3, 4$ 棵樹上的星星總和分別為 $3, 7, 12, 14$ 顆。

第 $3$ 到第 $4$ 棵樹間共有 $7$ 顆星星。

第 $1$ 到第 $3$ 棵樹間共有 $12$ 顆星星。

範例輸入 2 中，從左邊數來的前 $1, 2, 3$ 棵樹上的星星總和分別為 $3, 13, 20$ 顆。

第 $1$ 到第 $3$ 棵樹間共有 $20$ 顆星星。

第 $3$ 到第 $3$ 棵樹間共有 $7$ 顆星星。

範例輸入 3 中，從左邊數來的前 $1, 2, 3, 4, 5, 6$ 棵樹上的星星總和分別為 $99, 198, 297, 396, 495, 594$ 顆。

第 $1$ 到第 $1$ 棵樹間共有 $99$ 顆星星。

第 $3$ 到第 $4$ 棵樹間共有 $198$ 顆星星。

第 $5$ 到第 $6$ 棵樹間共有 $198$ 顆星星。

第 $1$ 到第 $6$ 棵樹間共有 $594$ 顆星星。

# Star Trees

***(10 points)***

Time Limit：1 second

Memory Limit：512 MB

## Description

Christmas is almost here! Thomas, Omelet, Joy and Eden came to the Christmas City.

As soon as you walk out of the train station, in addition to the crowded crowd, there is a row of Christmas trees full of stars! There are many stars on each tree, and curious Eden wants to know, for each $i$, how many stars are on the first $i$ trees starting from the left.

When he finally finished the calculation, Thomas asked him: Do you know how many stars there are in total between the $l$ tree and the $r$ tree?

Just when Eden was about to start counting again, the clever Omelet jumped out, looked at the calculation process that Eden had just done, and gave the answer directly! Thomas asked hundreds of times in a row, and Omelet could answer immediately.

Do you know how Omelet did it? Please help Eden figure out the answer to his question first, then try to imitate Omelet to quickly answer Thomas' question!

## Input Format

The first line of input has an integer $N$, representing a total of $N$ Christmas trees.

The second line has $N$ integers $a_i$ representing the number of stars in the $i$ th tree from left to right.

The third line has an integer $Q$, which means that Thomas asked $Q$ questions.

From line $4$ to line $Q+3$, each line has two integers $l, r$, representing that Thomas wants to know how many stars are there between $l$ th tree and $r$ th tree (including $l, r$).

## Output Format

Output $N$ integers on the first line, the $i$ th number represents the total number of stars on the first $i$ trees starting from the left.

For each line from line $2$ to line $Q+1$, output an integer representing the answer to Thomas's question.

## Constraints

$1 \leq N, Q \leq 500000$

$1 \leq a_i \leq 100$

$1 \leq l \leq r \leq N$

# Data Examples

## Input Example 1

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

## Output Example 1

```
3 7 12 14
7
12
```

## Input Example 2

```
3
3 10 7
2
1 3
3 3
```

## Output Example 2

```
3 13 20
20
7
```

## Input Example 3

```
6
99 99 99 99 99 99
4
1 1
3 4
5 6
1 6
```

## Output Example 3

```
99 198 297 396 495 594
99
198
198
594
```

# Example Explanation

In input example 1, the total number of stars on the first $1, 2, 3, 4$ trees starting from the left is $3, 7, 12, 14$ respectively.

There are $7$ stars between trees $3$ to $4$.

There are $12$ stars between trees $1$ to $3$.

In input example 2, the total number of stars on the first $1, 2, 3$ trees from the left is $3, 13, and 20$, respectively.

There are $20$ stars between trees $1$ to $3$.

There are $7$ stars between trees $3$ to $3$.

In input example 3, the total number of stars on the first $1, 2, 3, 4, 5, 6$ trees from the left is $99, 198, 297, 396, 495, 594$ respectively.

There are $99$ stars between trees $1$ to $1$.

There are $198$ stars between trees $3$ to $4$.

There are $198$ stars between trees $5$ to $6$.

There are $594$ stars between trees $1$ to $6$.

# Taiko Time

***(20分)***

## 問題敘述

「節奏躍然指上！」

osu! 是一款熱門的音樂遊戲，其目前有超過一千八百萬名玩家。osu! 一共有四種不同的遊戲模式：osu!、osu!taiko、osu!catch、osu!mania，於每年舉辦五場各模式的世界賽，並邀請知名音樂遊戲作曲家製作客製化樂曲，其冠軍賽總能吸引上千人同時觀看！

其中在 osu!taiko 的世界賽中，臺灣隊曾在 2011 與 2013 拿下冠軍，並在 2015 到 2019 年連續五年獲得亞軍，可說是臺灣人最擅長的遊戲模式！而在這道題目中，你的目標是寫一支程式進行簡化版的 osu!taiko 分數計算。若題目敘述中的內容與真實 osu!taiko 狀況不同，請以題目為準。


在 osu!taiko 模式中，各種音符會隨著音樂節奏從螢幕的右側向左移動，玩家必須在正確的時間點按下正確的按鍵以得到分數。在本題中，我們只考慮兩種音符：小的紅色音符（don，敲擊鼓面的聲音）和小的藍色音符（kat，敲擊鼓邊的聲音）。為了表示方便，玩家有時會以 `D` 表示紅色音符，`K` 表示藍色音符。

每個音符有其各自應該敲擊的時機點，設第 $i$ 個音符的時機點為第 $t_i$ 毫秒，且此音符的種類為 $b_i$。但要完美地在該時機點按下按鍵太困難了，因此在遊戲中，只要在指定的時間範圍內擊中音符即可獲得分數。此時間範圍由一個介於 $[0, 10]$ 的常數 $D$ 決定，其計算方式如下：先計算出三個用來進行判定的常數 $J_G, J_O, J_M$。其中 $J_G = \lfloor 49.5 - 3D \rfloor$；$J_O = \lfloor 79.5 - 8(D - 5) \rfloor$；$J_M$ 則依 $D$ 的大小有不同的計算方式，當 $D \leq 5$ 時，$J_M = \lfloor 94.5 - 8(D - 5) \rfloor$，當 $D > 5$ 時，$J_M = \lfloor 94.5 - 5(D - 5) \rfloor$。其中，$\lfloor x \rfloor$ 表示不超過 $x$ 的最大整數，例如 $\lfloor 7.27 \rfloor = 7$，$\lfloor 5.5 \rfloor = 5$。

假設玩家第 $j$ 次按下按鍵的時機點為 $s_j$ 毫秒，且敲擊的音符種類為 $a_j$。從第 1 個音符開始，遊戲會依序對每個音符進行判定。在判定第 $i$ 個音符時，系統會從所有玩家按下按鍵的時間點中選出至多一個符合以下所有條件的 $j$ 並以此作為音符 $i$ 判定的依據：

- 時間點 $j$ 尚未被其他音符作為判定的依據
- $t_i - J_M \leq s_j \leq t_i + J_M$

若不存在符合條件的時間點 $j$，則音符 $i$ 的判定為 MISS。若有兩個或以上的 $j$ 符合以上條件，則只選擇最小的 $j$ 對音符 $i$ 進行判定：若玩家敲擊的音符種類錯誤，也就是 $a_j \neq b_i$，則該音符的判定為 MISS。否則，系統將根據 $|t_i - s_j|$ 判定：若 $|t_i - s_j| \leq J_G$，則該音符的判定為 GREAT；若 $J_G < |t_i - s_j| \leq J_O$，則該音符的判定為 OK；若 $J_O < |t_i - s_j| \leq J_M$，則該音符的判定為 MISS。根據不同的判定，音符 $i$ 會得到不同的基本分 $v_i$：若判定為 GREAT，則基本分 $v_i = 300$；若判定為 OK，則基本分 $v_i = 150$；若判定為 MISS，則基本分 $v_i = 0$。

在完成了各音符的判定之後，接下來便是最重要的計分環節了！不過在計算分數之前，我們還須了解兩個名詞：Combo 和 Kiai Time。

Combo，也就是連擊數，指的是玩家連續擊出的 MISS 以外的判定的音符數量。對於音符 $i$，若它的判定為 MISS，則它的 Combo 數 $c_i = 0$；否則，$c_i = c_{i-1} + 1$。另外，我們定義 $c_0 = 0$，也就是若第 1 個音符的判定不是 MISS，則它的 Combo 數 $c_1 = 1$。

接下來是 Kiai Time，通常 Kiai Time 用來表示樂曲想要強調的部分。在 Kiai Time 期間，得分是非 Kiai Time 的 $1.2$ 倍！假設樂曲的第 $h$ 段 Kiai Time 為 $[l_h, r_h]$ 毫秒，則當音符 $i$ 位於任意一段 Kiai Time，也就是存在任意一個 $h$ 使得 $l_h \leq t_i \leq r_h$ 時，其 $k_i = 1.2$；否則，其 $k_i = 1$。

根據以上計算的數值，對於每個音符 $i$，我們可以算出其最終的分數 $p_i$，其計算公式如下：
$p_i = [v_i + \min(\lfloor \frac{\max(c_i - 1, 0)}{10} \rfloor, 10) \times 80)] \times k_i$，其中 $\min(x, y)$ 表示 $x$ 和 $y$ 之中較小的數，$\max(x, y)$ 表示 $x$ 和 $y$ 之中較大的數。最後，將所有 $p_i$ 加總，即可得到本次遊玩的得分了！

在結算畫面中，除了得分以外，系統還會另外顯示一些統計。首先是三種判定的音符數量：判定為 GREAT 的音符數量 $G$、判定為 OK 的音符數量 $O$、判定為 MISS 的音符數量 $M$，以及 Accuracy 和 Max Combo。Accuracy 表示玩家的準確率，其計算方式為 $\frac{G + 0.5O}{G + O + M}$，Max Combo 則是玩家累積的最大 Combo 數，也就是所有音符中 $c_i$ 最大者。

喜愛 osu! 熱愛 osu! 沒有 osu! 就吃不下飯就睡不著覺的你，為了成為 2023 年的 osu!taiko 世界賽選手，請寫一支程式進行 osu!taiko 的分數計算吧！

# 輸入格式

輸入的第一行包含四個數 $N, H, D, K$，分別表示音符數量、玩家按下按鍵的次數、決定判定時間範圍的常數、Kiai Time 的數量。其中，$N, H, K$ 為整數，$D$ 為實數，其小數點後至多有兩位。

第二行包含 $N$ 個整數 $t_i$，表示各音符的時機點；第三行包含一個長度為 $N$ 的字串 $b$，表示第 $i$ 個音符的種類。

第四行包含 $H$ 個整數 $s_j$，表示玩家按下按鍵的時機點；第五行包含一個長度為 $H$ 的字串 $a$，表示玩家敲擊的第 $j$ 個按鍵的音符種類。

接下來 $K$ 行，每行包含兩個整數 $l_h, r_h$，表示第 $h$ 段 Kiai Time 的開始與結束時間。

# 輸出格式

輸出的第一行包含一個整數，表示本次遊玩的得分。第二行包含三個整數 $G, O, M$，表示三種判定的音符數量。第三行請輸出玩家的準確率的百分比，並四捨五入至小數點後第二位。第四行請輸出一個整數，表示玩家的累積的最大 Combo 數。

# 資料範圍

- $1 \leq N \leq 65536$
- $0 \leq H \leq 10^5$
- $0 \leq D \leq 10$
- $0 \leq K \leq 10^5$
- $1 \leq t_i, s_j \leq 10^9$
- $\forall 1 < i \leq N, t_{i-1} < t_i$
- $\forall 1 < j \leq H, s_{j-1} < s_j$
- $a_i, b_j \in \{\text{'D'}, \text{'K'}\}$
- $1 \leq l_h \leq r_h \leq 10^9$
- $\forall 1 < h \leq K, r_{h-1} < l_h$

# 資料範例

## 輸入範例 1

```
7 8 5 1
200 400 600 1000 1100 1450 2434
DDKKDKK
100 106 200 399 650 1050 1116 1488
DDDDKKDD
400 1000
```

## 輸出範例 1

```
1020
2 2 3
42.86
4
```

## 輸入範例 2

```
16 16 9.87 2
100 200 300 400 500 600 700 800 900 1000 1100 1200 1300 1400 1500 1600
DDKKDKDKDKDDDDDK
100 200 300 400 500 600 700 800 900 1000 1100 1200 1300 1400 1500 1600
DKKKDKDKDKDDDDKK
200 300
1400 1600
```

## 輸出範例 2

```
4556
14 0 2
87.50
12
```

## 輸入範例 3

```
6 6 4.01 0
100 300 400 500 600 727
DDDKKK
100 200 300 400 500 600
DDDKKK
```

## 輸出範例 3

```
300
1 0 5
16.67
1
```

# 範例說明

範例測資 1 中，前 $6$ 個音符對應到的玩家敲擊的時間點 $j$ 為 $2, 4, 5, 6, 7, 8$，其中第 $6$ 個音符對應到的敲擊種類是錯誤的，而第 $7$ 個音符沒有對應到任何 $j$，判定依序為 MISS、GREAT、OK、OK、GREAT、MISS、MISS。$v_i$ 依序為 $0, 300, 150, 150, 300, 0, 0$。第 $2$ 到第 $4$ 個音符在 Kiai Time 期間，故 $k_2 = k_3 = k_4 = 1.2$；其餘音符 $k_i = 1$。$p_i$ 則依序為 $0, 360, 180, 180, 300, 0, 0$，總分為 $1020$。

範例輸入 2 中，第 $2$ 和第 $15$ 個音符判定為 MISS，其餘音符判定為 GREAT。

範例輸入 3 中，第 $1$ 個音符判定為 GREAT，其餘音符判定為 MISS。

# Taiko Time

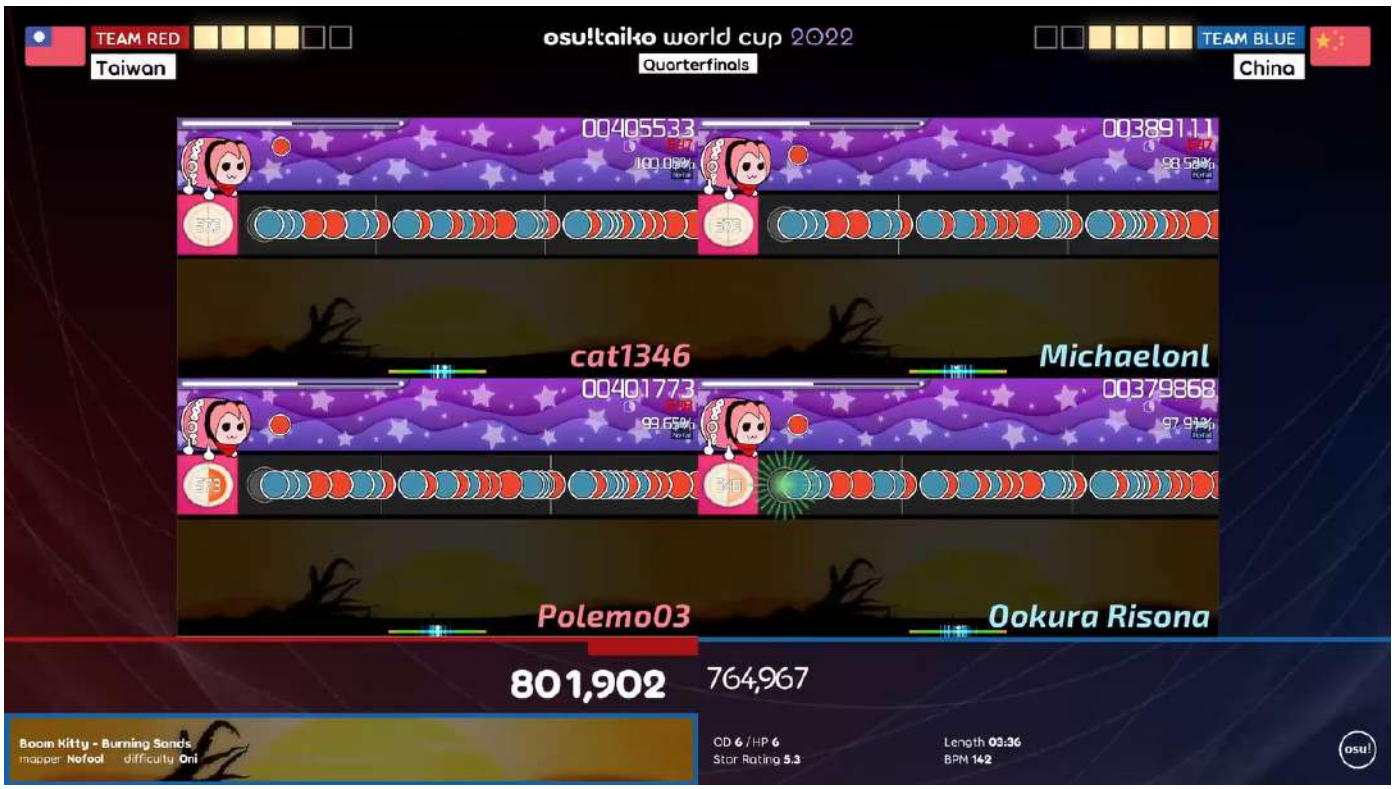
***(20 Points)***

## Description

"Rhythm is just a click away!"

osu! is a popular rhythm game which has over 18 million players. It consists of four different game modes: osu!, osu!taiko, osu!catch, and osu!mania. There are five world cup hosted by osu! team for these game modes each year, and many featured artists are invited to compose custom songs for the tournaments. Thousands of viewers will crowd into osu!'s official twitch channel to watch the grand finals matches!

Team Taiwan won the osu!taiko World Cup in 2011 and 2013, and they also getting the second place for five consecutive years from 2015 to 2019. osu!taiko is one of the most popular game modes in Taiwan! In this problem, you're going to write a program to simulate how osu! calculate the score in osu!taiko mode. Please follow the problem description if there are some differnces with the real osu!taiko scoring system.




In osu!taiko, notes move leftward from the right side of the screen. Players have to press down the correct keys at the correct moment to earn scores. In this problem, we introduce only two kind of notes in the game - small red note (don, the sound of hitting the center of a drum), and small blue note (kat, the sound of hitting the edge of a drum). As a matter of convenience, letters `D` and `K` usually stand for small red notes and small blue notes respectively.

Each note has its own timing point to hit, let the $i$-th note's timing point be the $t_i$-th milisecond and its type be $b_i$. It is too difficult to hit the notes exactly at their timing points, therefore, players only need to hit the notes in a specific interval of time to earn scores. The interval is determined by a real number $D \in [0, 10]$. We first calculate three constants - $J_G, J_O, J_M$, where $J_G = \lfloor 49.5 - 3D \rfloor$, $J_O = \lfloor 79.5 - 8(D - 5) \rfloor$. The

formula for $J_M$ varies for different $D$, $J_M = \lfloor 94.5 - 8(D-5) \rfloor$ when $D \leq 5$, and $J_M = \lfloor 94.5 - 5(D-5) \rfloor$ when $D > 5$. $\lfloor x \rfloor$ stands for the floor function, which returns the greatest interger less than or equal to $x$. For example, $\lfloor 7.27 \rfloor = 7$, $\lfloor 5.5 \rfloor = 5$.

Let the timing point of the $j$-th key press from the player be the $s_j$-th milisecond and its type be $a_i$. The game gives a judgement for each note chronologically from the first note. When judging the $i$-th note, the game select at most one key press $j$ from the player to give a judgement that satisfies both conditions below:

- $j$-th key press is not used to judge any notes before
- $t_i - J_M \leq s_j \leq t_i + J_M$

If there are no key press satisfies both conditions above, the judgement of the $i$-th note will be a MISS. If there are two or more key presses satisfy both conditions above, the game only chooses the one with the smallest $j$ and use it to judge the $i$-th note. If the player is hitting the wrong type of note, which means $a_j \neq b_i$, the judgement of the $i$-th note will be a MISS. Otherwise, the game judge the note by the value of $|t_i - s_j|$. If $|t_i - s_j| \leq J_G$, the judgement of the $i$-th note will be a GREAT. If $J_G < |t_i - s_j| \leq J_O$, the judgement of the $i$-th note will be an OK. If $J_O < |t_i - s_j| \leq J_M$, the judgement of the $i$-th note will be a MISS. The $i$-th note has its basic score value $v_i$ according to its judgement. The basic score value for the $i$-th note $v_i$ will be $300$, $150$, or $0$ respectively if its judgement is GREAT, OK, or MISS.

After giving the judgement for every note, we can finally start to deal with the total score! Before kicking start to calculate, we still have to understand to special terms in osu!taiko - 'Combo' and 'Kiai Time.'

Combo means the consecutive notes that the player has hit without getting a MISS judgement. Formally, for the $i$-th note, let its Combo be $c_i$. If the judgement of the $i$-th note is a MISS, its Combo $c_i$ is $0$; otherwise, its Combo $c_i$ is $c_{i-1} + 1$. Besides, we define that $c_0 = 0$, which means the Combo of the first note $c_1$ is $1$ if its judgement is not a MISS.

Kiai Time is used to emphasize some part of the music. Players earn 20% more scores during the Kiai Time! Let the $h$-th Kiai Time of the music be $[l_h, r_h]$ milisecond. If the $i$-th note is in any of the Kiai Time, which means there exist a $h$ such that $l_h \leq t_i \leq r_h$, the Kiai Multiplier of the $i$-th note $k_i$ is $1.2$; otherwise, $k_i$ is $1$.

We can calculate the final score of the $i$-th note by the values we obtain above. Let $p_i$ be the final score of the $i$-th note, $p_i = [v_i + \min(\lfloor \frac{\max(c_i - 1, 0)}{10} \rfloor, 10) \times 80)] \times k_i$, where $\min(a, b)$ denotes the smaller number between $a$ and $b$, and $\max(a, b)$ denotes the bigger number between $a$ and $b$. Summing up the final score for all the notes - and that's the total score of the gameplay!

Besides the total score, the game also gives some other statistics to on the result screen. First of all, the number of notes with a GREAT judgement $G$, with an OK judgement $O$, with a MISS judgement $M$. Accuracy and Max Combo is also shown on the result screen. We calculate the value of Accuracy with the formula $\frac{G + 0.5O}{G + O + M}$, and the value of Max Combo is the maximal value among all $c_i$.

You, as an osu! enthusiast, in order to prepare to participate in osu!taiko World Cup 2023, are going to write a program to calculate the score of an osu!taiko gameplay!

# Input Format

The first line contains four numbers $N, H, D, K$ - the number of notes, the number of keypresses, the real number to determine the judgement interval, the number of Kiai Time, where $N, H, K$ are intergers, and $D$ is a real number with at most two decimal places.

The second line contains $N$ integers $t_i$ - the timing points for each note. The third line is the string $b$ of length $N$ - the type of each note.

The fourth line contains $H$ integers $s_j$ - the timing points for each key press. The fifth line is the string $a$ of length $H$ - the type of each key press.

Each of the next $K$ lines contains two interger $l_h, r_h$ - the interval of the $h$-th Kiai Time.

# Output Format

The first line should contain an integer - the total score of the gameplay. The second line should contain three intergers $G, O, M$ - the number of notes with a GREAT, OK, MISS judgement. The third line should contain a real number - the value of Accuracy in percantage, rounded to two decimal places. The fourth line should contain an integer - the value of Max Combo.

# Constraints

- $1 \leq N \leq 65536$
- $0 \leq H \leq 10^5$
- $0 \leq D \leq 10$
- $0 \leq K \leq 10^5$
- $1 \leq t_i, s_j \leq 10^9$
- $\forall 1 < i \leq N, t_{i-1} < t_i$
- $\forall 1 < j \leq H, s_{j-1} < s_j$
- $a_i, b_j \in \{\text{'D'}, \text{'K'}\}$
- $1 \leq l_h \leq r_h \leq 10^9$
- $\forall 1 < h \leq K, r_{h-1} < l_h$

## Input Example 1

```
7 8 5 1
200 400 600 1000 1100 1450 2434
DDKKDKK
100 106 200 399 650 1050 1116 1488
DDDDKKDD
400 1000
```

## Output Example 1

```
1020
2 2 3
42.86
4
```

## Input Example 2

```
16 16 9.87 2
100 200 300 400 500 600 700 800 900 1000 1100 1200 1300 1400 1500 1600
DDKKDKDKDKDDDDDK
100 200 300 400 500 600 700 800 900 1000 1100 1200 1300 1400 1500 1600
DKKKDKDKDKDDDDKK
200 300
1400 1600
```

## Output Example 2

```
4556
14 0 2
87.50
12
```

## Input Example 3

```
6 6 4.01 0
100 300 400 500 600 727
DDDKKK
100 200 300 400 500 600
DDDKKK
```

## Output Example 3

```
300
1 0 5
16.67
1
```

# Example Explanation

For sample test case 1, the key presses used to judge the first $6$ note are the $2$nd, $4$th, $5$th, $6$th, $7$th, $8$th, respectively. The $6$th note is hit with a wrong type of key press, and there's no key press to judge the $7$th note. The judgement for the notes are MISS, GREAT, OK, OK, GREAT, MISS, MISS, and the basic score value $v_i$ are $0, 300, 150, 150, 300, 0, 0$, respectively. The $2$nd to the $4$th note are in the Kiai Time, therefore $k_2 = k_3 = k_4 = 1.2$. All the other notes have the Kiai Multiplier $k_i = 1$. The final score of the notes $p_i$ are $0, 360, 180, 180, 300, 0, 0$, the total score is $1020$.

For sample test case 2, the $2$nd and the $15$th notes are judged as MISS, all the other notes are judged as GREAT.

For sample test case 3, the $1$st note is judged as GREAT, all the other notes are judged as MISS.

# P力量 - P Power

***(20分)***

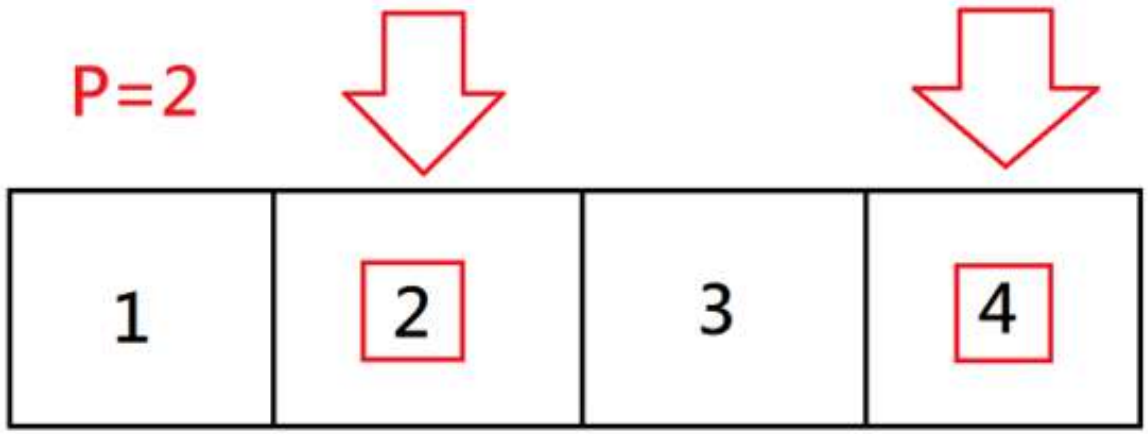
時間限制: 4 秒
記憶體限制: 512 MB

## 問題敘述

一個序列與一個質數之間會產生一股力量，稱之為$P$力量。
若序列中有$K$個數字能夠被那個質數整除的話，則它們之間的$P$力量就是$K$。

史密斯是一名專攻$P$力量的大博士，有一天他發現了一個序列$S$。
為了研究這個序列，史密斯決定調查這個序列的一些連續子序列，$S[L_1, R_1], S[L_2, R_2], \cdots, S[L_Q, R_Q]$ 與某些質數 $P_1, P_2, \ldots, P_Q$ 之間產生的力量為多少。
為了方便起見，這些子序列的左界與右界分別都會單調遞增。
也就是說，$L_1 \leq L_2 \leq \ldots \leq L_{Q-1} \leq L_Q$ 且 $R_1 \leq R_2 \leq \ldots \leq R_{Q-1} \leq R_Q$。

以下圖為例：




序列 $(1, 2, 3, 4)$ 與質數2之間的$P$力量為2，因為這個序列中有兩個元素可以被2整除。

## 輸入格式

輸入的第一行會包含一個整數 $T$，代表測資數量。
每筆測資的第一行會有兩個整數 $N, Q$，代表序列的長度與詢問的數量。
每筆測資的第二行會有 $N$ 個整數 $S_1, S_2, \cdots, S_N$，代表序列中的元素。
每筆測資接下來的 $Q$ 行，每行會有三個整數 $L_i, R_i, P_i$，代表史密斯想要知道 $S[L_i, R_i]$ 所構成的子序列與 $P_i$ 之間的$P$力量為多少。

# 輸出格式

對於每筆詢問，輸出一個整數代表那個子序列與那個質數之間的$P$力量。

# 資料範圍

- $1 \leq T \leq 10^3$
- $1 \leq N, Q \leq 5 \times 10^5$
- 保證所有測資中的$N$加總不會超過 $5 \times 10^5$
- 保證所有測資中的$Q$加總不會超過 $5 \times 10^5$
- $\forall i = 1, 2, \cdots, N$，$1 \leq S_i \leq 5 \times 10^5$
- $\forall i = 1, 2, \cdots, Q$，$1 \leq L_i \leq R_i \leq N$，$1 \leq P_i \leq 5 \times 10^5$
- $\forall i = 1, 2, \cdots, Q - 1$，$L_i \leq L_{i+1}$，$R_i \leq R_{i+1}$

# 資料範例

## 輸入範例 1

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

## 輸出範例 1

```
1
1
2
0
1
```

## 輸入範例 2

```
2
5 3
2 2 2 2 2
1 1 2
1 3 3
2 5 7
4 3
3 6 9 12
1 2 2
2 3 3
2 4 2
```

## 輸出範例 2

```
1
0
0
1
2
2
```

## 輸入範例 3

```
1
6 5
2 6 30 210 2310 30030
2 3 3
3 4 5
4 5 7
5 6 11
6 6 13
```

## 輸出範例 3

```
2
2
2
2
1
```

# 範例說明

範例一中

- 在 $S[1,2]$ 中$S[2]$可以被$2$整除，因此輸出$1$。
- 在 $S[1,3]$ 中$S[3]$可以被$3$整除，因此輸出$1$。

- 在 $S[1,4]$ 中$S[2], S[4]$可以被$2$整除，因此輸出$2$。
- 在 $S[2,4]$ 中沒有元素可以被$5$整除，因此輸出$0$。
- 在 $S[3,4]$ 中$S[4]$可以被$2$整除，因此輸出$1$。

# P Power

***(20 points)***

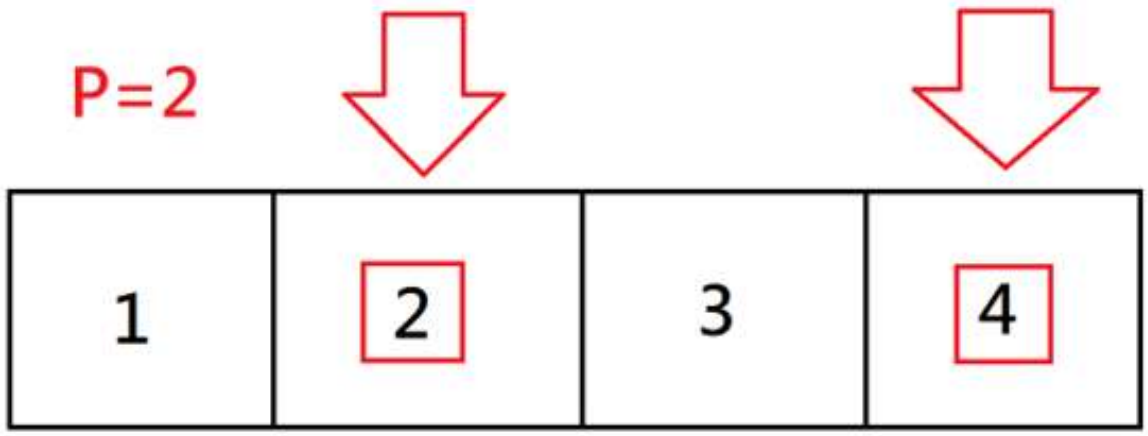
Time Limit: 4 seconds
Memory Limit: 512 MB

## Description

There is some power between a sequence and a prime number, called $P$ power.
If there are $K$ elements which can be divided by that prime number in the sequence, then the $P$ power between them is $K$.

Smith is a professor who researches $P$ power. One day he finds a sequence $S$.
To research this sequence, Smith decides to find the $P$ power between some consecutive subsequence, $S[L_1, R_1], S[L_2, R_2], \cdots, S[L_Q, R_Q]$ and some prime numbers $P_1, P_2, \ldots, P_Q$.

For convenience, it is guaranteed that $L_1 \leq L_2 \leq \ldots \leq L_{Q-1} \leq L_Q$ and $R_1 \leq R_2 \leq \ldots \leq R_{Q-1} \leq R_Q$.




The $P$ power between the sequence $(1, 2, 3, 4)$ and the prime number $2$ is 2, since there are two elements in the sequence which can be divided by $2$.

## Input Format

The first line of the input contains an integer $T$, representing the number of test cases.
The first line of each test case contains two integers $N, Q$, representing the length of the sequence and the number of queries.
The second line of each test case contains $N$ integers, $S_1, S_2, \ldots, S_N$,representing the elements in the sequence.
Each of next $Q$ line contains three integers $L_i, R_i, P_i$, representing that Smith wants to know the $P$ power between the subsequence $S[L_i, R_i]$ and $P_i$.

# Output Format

For each query, please output the $P$ power between the subsequence and the prime number.

# Constraints

- $1 \leq T \leq 10^3$
- $1 \leq N, Q \leq 5 \times 10^5$
- It is guaranteed the sum of $N$ doesn't exceed $5 \times 10^5$.
- It is guaranteed the sum of $Q$ doesn't exceed $5 \times 10^5$.
- $\forall i = 1, 2, \cdots, N$ , $1 \leq S_i \leq 5 \times 10^5$
- $\forall i = 1, 2, \cdots, Q$ , $1 \leq L_i \leq R_i \leq N$ , $1 \leq P_i \leq 5 \times 10^5$
- $\forall i = 1, 2, \cdots, Q - 1$ , $L_i \leq L_{i+1}$ , $R_i \leq R_{i+1}$

## Input Example 1

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

## Output Example 1

```
1
1
2
0
1
```

## Input Example 2

```
2
5 3
2 2 2 2 2
1 1 2
1 3 3
2 5 7
4 3
3 6 9 12
1 2 2
2 3 3
2 4 2
```

## Output Example 2

```
1
0
0
1
2
2
```

## Input Example 3

```
1
6 5
2 6 30 210 2310 30030
2 3 3
3 4 5
4 5 7
5 6 11
6 6 13
```

## Output Example 3

```
2
2
2
2
1
```

# Example Explanation

In example 1.

- $S[2]$ in $S[1, 2]$ can be divided by $2$, so you should print $1$.
- $S[3]$ in $S[1, 3]$ can be divided by $3$, so you should print $1$.
- $S[2], S[4]$ in $S[1, 4]$ can be divided by $2$, so you should print $2$.
- No element in $S[2, 4]$ can be divided by $5$, so you should print $0$.
- $S[4]$ in $S[3, 4]$ can be divided by $2$, so you should print $1$.