Concurso de Programación

Concurso I 2026 - Estructura de Datos

https://urjc-cp.github.io/urjc-cp/

Cuadernillo de problemas

Realizado en la **Escuela Técnica Superior de Ingeniería Informática (URJC)**
20 de febrero de 2026

Índice

A Stack Overflow **3**

B A veces, César **4**

C Chuches Gratis **5**

D Artículo en Revisión **6**

E Envíos Prioritarios **7**

F Star Gazer **8**

Autores de los problemas:

- Iván Penedo Ventosa (Habla Computing, Universidad Rey Juan Carlos)
- Adaya Ruiz Mayoral (GMV, Universidad Rey Juan Carlos)
- Lucas Martín García (Universidad Rey Juan Carlos)
- Sara García Rodríguez (Universidad Rey Juan Carlos)
- Isaac Lozano Osorio (Universidad Rey Juan Carlos)
- Olga Somalo Serrano (Universidad Rey Juan Carlos)

Tiempo: 0.2 segundo(s)

A
Stack Overflow

Como estudiante de informática que eres, siempre habías conocido StackOverflow como aquel Santo Grial al que acudir cuando el código de la práctica de Introducción a la Programación hacía algo raro. Pero este año, mientras estudiabas Estructura de Datos, te has encontrado con que este término significa algo más. Te has dado cuenta de que se refiere a un error nuevo que nunca antes habías visto, el cual viene acompañado de una traza que te indica los métodos por los que ha estado navegando tu código.

Investigando, te has dado cuenta de que esto tiene que ver con lo que se conoce como *call stack* en inglés, la cual es una estructura de datos utilizada por los lenguajes de programación para realizar un seguimiento de las funciones que se están ejecutando en un programa. Cada vez que se invoca una función, esta se almacena, y cuando una función termina su ejecución, se elimina. Esto permite al programa recordar el punto en el que se encontraba antes de llamar a la función, y al mismo tiempo, gestionar las dependencias entre las distintas funciones.

Sin embargo, Java a veces hace cosas raras, y ahora te está generando una lista con las llamadas que se van a realizar, además de indicar cuándo finaliza el método que se ejecutaría en ese momento. Por lo tanto, te va a tocar encontrar en cuál de las ejecuciones tu código va a fallar y cuál es la traza hasta ese punto.

Entrada

La entrada comenzará con dos números enteros N y M separados por un espacio. N indica la capacidad máxima de la pila, mientras que M el número de líneas de eventos a tratar. A continuación, aparecerán M líneas indicando las llamadas realizadas con el formato `nombre_de_funcion:línea`, o `RETURN` si se ha finalizado la última función indicada.

El de las funciones estarán formadas por caracteres del alfabeto inglés y barras bajas "_".

Salida

En la salida se indicará `Exited with code 0` si el programa se ha ejecutado correctamente. Si por el contrario se ha llenado el *stack*, se deberá mostrar la traza inversa del programa cuando haya saltado la excepción, de manera que la primera línea contendrá la función y la línea que ha dado fallo.

Independientemente de que la lista de llamadas que se te de tenga más funciones para ejecutar, el programa fallará en el momento en el que se alcance la capacidad máxima del *stack* y, por lo tanto, el resto de la entrada se puede ignorar, pero se deberá leer por completo.

Entrada de ejemplo

```
3 6
main:1
hello_world:3
RETURN
do_something:4
ups:15
print:6
```

Salida de ejemplo

```
ups:15
do_something:4
main:1
```

Límites

- $0 \leq N, M \leq 200000$

Tiempo: 1 segundo

B
A veces, César

En la antigua Roma, los mensajes secretos eran una herramienta fundamental para mantener el control del Imperio. Uno de los métodos más conocidos era el cifrado atribuido a Julio César, popularizado hoy en día en su variante ROT13: cada letra del mensaje se desplazaba un número fijo de posiciones en el alfabeto.

Sin embargo, los historiadores han descubierto que el César no siempre utilizaba el mismo desplazamiento. En ocasiones, cambiaba la rotación según la urgencia del mensaje, el destinatario o simplemente su estado de ánimo.

Tu misión es descifrar mensajes imperiales. Para ayudarte, el Senado te proporciona un ejemplo: una palabra cifrada y su correspondiente versión descifrada. A partir de ese ejemplo deberás determinar cuántas posiciones se rotó el alfabeto. Luego, deberás aplicar esa misma rotación para descifrar un nuevo mensaje interceptado.

Para este problema se considerará el alfabeto inglés de 26 letras minúsculas `abcdefghijklmnopqrstuvwxyz`, donde la ñ no forma parte del alfabeto. El cifrado consiste en desplazar cada letra N posiciones hacia delante en el alfabeto. Si se supera la z, se continúa desde la a. Para descifrar, se desplaza N posiciones hacia atrás.

Entrada

La entrada de este problema estará formada por varios casos de prueba donde cada caso de prueba corresponde a un tipo de rotación, hasta final de fichero.

Cada caso de prueba está compuesto por 3 líneas. Para cada caso de prueba la primera línea estará formada por una palabra cifrada $C1$. En la siguiente línea se encontrará esa misma palabra descifrada $D1$. Por último, en la tercera línea estará la palabra cifrada de la misma forma que la anterior $C2$, y que se debe descifrar. Se garantiza que $C1$ y $D1$ tienen la misma longitud, y que existe un único valor de N consistente en el ejemplo dado.

Salida

Para cada caso de prueba, deberás devolver una única línea con la palabra descifrada $D2$ en minúscula.

Entrada de ejemplo

```
krod
hola
pxqgr
bdrzq
cesar
dlodqzcnq
```

Salida de ejemplo

```
mundo
emperador
```

Límites

- $1 \leq |C1| = |D1| \leq 10^5$
- $1 \leq |C2| \leq 10^5$

Tiempo: 1 segundo

C
Chuches Gratis

Una empresa está repartiendo chuches gratis a la salida de la universidad. Es sencillo: vas, les das tus datos y te dan tus chuches. Sin embargo, los repartidores empiezan a notar que algunas caras les suenan, están repitiendo. La empresa no puede permitir tal cosa. Quiere darte las chuches solo la primera vez.

Entrada

La entrada corresponde a un único caso de prueba. La primera línea es un entero P, que indica cuántas personas han venido a pedir chuches. Le siguen P líneas, cada una de ellas con el nombre de la persona que pide. El nombre es una única palabra en minúsculas y con caracteres del alfabeto inglés.

Salida

La salida consistirá en P líneas, una por cada persona que viene a pedir chuches, escribiendo "CHUCHE" si es la primera vez que viene o "REPE" si ha venido ya. Se asume que por muy grande que sea la universidad no va a haber 2 personas con el mismo nombre, si se llama igual es el mismo.

Entrada de ejemplo

```
11
alicia
isaac
olga
alicia
ivan
adaya
sergio
lucas
alicia
sara
alicia
```

Salida de ejemplo

```
CHUCHE
CHUCHE
CHUCHE
REPE
CHUCHE
CHUCHE
CHUCHE
CHUCHE
REPE
CHUCHE
REPE
```

Límites

- $1 \leq P \leq 10^6$

Tiempo: 1 segundo

D
Artículo en Revisión

Aunque parezca mentira los profesores de la universidad también tienen que sacar tiempo para investigar, pues es uno de los requisitos para optar a nuevas plazas e incentivos. Pero... ¿qué es esto de investigar? Investigar realmente lleva un gran trabajo por detrás: seleccionar una temática, estudiar todos los trabajos previos, analizar los resultados, proponer nuevas contribuciones, demostrarlas, realizar un documento científico y enviar a una revista con prestigio para que se valore. Según la Agencia Nacional de Evaluación de la Calidad y Acreditación (ANECA), el tiempo promedio de que un artículo de investigación sea aceptado (sin contar el trabajo que tiene por detrás) es de 142 días.

Las primeras personas que leen los artículos enviados a la revista son los editores. Estos editores están encargados de asociar revisores a los artículos que reciban. Los revisores reportarán un feedback y el editor notificará a los autores con el veredicto. Los veredictos más comunes, ordenados de mejor a peor, son los siguientes: aceptado (AC), cambios menores (Cm), cambios mayores (CM) o rechazado (RJ). Dado el gran volumen de artículos recibidos los tiempos de espera para obtener una respuesta por parte de los investigadores es realmente extenso. ¿Crees que podrías echar una mano a los editores de las revistas? Tú misión será, dado el veredicto de un conjunto de revisores, reportar el peor de todos ellos.

Entrada

Existirá un único caso de prueba. Este caso de prueba tendrá un número R con el total de revisores que han reportado feedback. Por cada revisor se mostrará en una nueva línea su propuesta: “AC”, “Cm”, “CM” o “RJ”.

Salida

Para cada caso de prueba se mostrará el resultado de las propuestas de los revisores.

Entrada de ejemplo

```
4
AC
Cm
CM
RJ
```

Salida de ejemplo

```
RJ
```

Límites

- $1 \leq R \leq 4$

Tiempo: 1 segundo

E
Envíos Prioritarios

Amazon está pensando añadir un nuevo tipo de subscripción a su servicio de compras online. El nuevo servicio lleva las siglas de "P2W", nadie está seguro del significado que llevan detrás.
Los usuarios que opten por pagar los 100 euros que cuesta podrán hacer que sus envíos se salten posiciones en la cola de salida de los centros de reparto; en concreto, saldrán antes que cualquier producto que no se haya pedido con este servicio.

Entrada

La entrada contendrá un único caso de prueba.
El caso empezará con un número, N, que indicará el número de eventos que sucederán a lo largo de una semana en el centro de reparto. Cada evento ocupará una línea y puede tener uno de los siguientes formatos:

- "P2W" o "NOR" - indicando la llegada de un paquete "P2W" o normal a la cola de reparto.
- "R C" - indicando que va a salir un camión con capacidad para C paquetes del centro de reparto.

Tanto en la cola de reparto como en el camión los paquetes se ordenan por ser "P2W" o no y luego por su orden de aparición.

Salida

Por cada camión que sale tienes que imprimir los identificadores de los paquetes que se va a llevar el camión (sin rebasar su capacidad), separados por un espacio. El camión puede salir sin llenarse e incluso vacío si no hay paquetes suficientes esperando en la cola. Los paquetes se identifican por su orden de aparición en la entrada, empezando en el 0.

Entrada de ejemplo

```
8
NOR
NOR
P2W
NOR
R 2
P2W
R 5
NOR
```

Salida de ejemplo

```
2 0
4 1 3
```

Límites

- $1 \leq N \leq 300000$
- $1 \leq C \leq 100$

Tiempo: 1 segundo

F
Star Gazer

Ricky creció en un pequeño pueblo cerca del río Caraldulla hace mucho tiempo. Ricky nos contó que uno de sus momentos favoritos del día era cuando terminaba todas sus tareas, iba al río, se tumbaba en la hierba y observaba el cielo infinito buscando cometas. Cada vez que veía uno, pedía un deseo con todas sus fuerzas.

Algunos deseos se cumplieron, otros no. Ricky atribuía la falta de "suerte" únicamente a la fuerza del cometa, pensaba que si el cometa fuera más grande o incluso estuviera acompañado por otro cometa, el deseo podría hacerse realidad. Durante su vida, se las ha arreglado para obtener una lista de todos los cometas que ha visto y la fecha en que aparecieron por última vez. Ha hecho todos los cálculos y sabe cuánto tarda en pasar cada cometa.

Justo hoy se han alineado todos los cometas, pero como tenía ganas de ir a la clase de programación competitiva y ganar a sus amigos en el concurso, no va a poder pedir el deseo, por eso quiere saber cuándo será la próxima vez que se alineen para pedir el deseo. ¿Puedes ayudarle a saber cuándo ocurrirá esto?

Entrada

La entrada comienza con el número de casos de prueba, T. A continuación, seguirán las descripciones de los T casos de prueba. Para cada caso de prueba habrá dos líneas: la primera linea contiene un entero N denotando el número de cometas y luego, la segunda linea, N enteros, n_i, separados por un único espacio denotando la duración en meses que tarda en pasar el cometa i.

Salida

Para cada caso debes imprimir el tiempo (en meses) en el que todos los cometas estarán alineados y pasarán por el cielo para que Ricky pueda pedir un deseo con fuerza.

Entrada de ejemplo

```
3
4
1 2 3 4
4
2 4 8 64
7
2 3 5 7 11 13 17
```

Salida de ejemplo

```
12
64
510510
```

Límites

- $1 \leq T \leq 100$
- $1 \leq N \leq 100$
- $1 \leq n_i \leq 40$