**Barrancas del cobre**

Tiempo límite por caso: 0.5 seg

Memoria límite por caso: 32 MB

Las barrancas del cobre son uno de los puntos de interés turístico más famosos de Chihuahua. Una agencia de viajes te ha pedido ayuda para diseñar un recorrido a través de ellas.

El recorrido a través de las barrancas tiene ***N*** vistas espectaculares numeradas de ***1*** a ***N***. Por supuesto, hay algunas más llamativas que otras. Los expertos de la agencia han calificado cada una de las ***N*** vistas de acuerdo a su *espectacularidad*.

Los recorridos que vende la agencia llevan a los turistas a un punto de inicio y a partir de ahí visitan las siguientes ***K*** vistas. La agencia tiene una preocupación. Cuando llevas un turista de una vista a la siguiente, si la segunda vista es **más espectacular** el turista se siente **emocionado**, sin embargo si la segunda vista es **menos espectacular** el turista se siente **decepcionado**.

La emoción y decepción del turista pueden medirse. Cuando el turista pasa de una vista a la siguiente, su *emoción total* cambia de acuerdo a las siguientes reglas:

* Si la nueva vista es **más espectacular** la *emoción total* **aumenta** una cantidad igual a la diferencia de espectacularidad entre ambas barrancas.
* Si la nueva vista es **menos espectacular** la *emoción total* **disminuye**una cantidad igual a la diferencia de espectacularidad entre ambas barrancas.

Obviamente la agencia no puede reacomodar las barrancas, así que en un recorrido el turista pasará por vistas que aumenten su emoción y vistas que disminuyan su emoción.

La agencia quiere que le ayudes a encontrar el recorrido con el cual el turista quedará más emocionado.

**NOTA: Te recomendamos ampliamente revisar la explicación del ejemplo para que el problema te quede más claro.**

**Problema**

Escribe un programa que dada la espectacularidad de cada una de las ***N*** vistas y el número ***K*** de vistas que se van a incluir en el recorrido encuentre cuál es la emoción total máxima que se puede lograr.

**Entrada**

Tu programa debe leer del teclado la siguiente información:

* En la primera línea los número ***N*** y ***K***, el número de vistas y la longitud del recorrido.
* En la siguiente línea habrá ***N*** números no negativos separados por un espacio que indican la espectacularidad de las ***N*** vistas.

**Salida**

Tu programa debe escribir en la pantalla un único número que representa la *emoción* *total* máxima posible para un recorrido de ***K*** vistas.

**Restricciones**

**2 <= *N* <= 50,000** Número de vistas.

**2 <= *K* <= *N*** Cantidad de vistas en el recorrido.

**0 <= *ei* <= 10,000** Espectacularidad de la ***i-***ésima vista.

Si te da WA, ¡corrígelo!

**Ejemplo**

|  |  |  |
| --- | --- | --- |
| **Entrada**  | **Salida**  | **Explicación** |
| 6 42 1 3 2 6 2 | 5 | El recorrido será de 4 vistas. Si se inicia en la primera vista, la *emoción total* se calcula como sigue:* De la primera vista a la segunda se disminuye la espectacularidad. La emoción total **disminuye** por 1 porque *2–1=1* y la emoción total va en *-1*.
* De la segunda a la tercera la espectacularidad **aumenta**. La emoción total aumenta por 2 porque *3-1=2*. La emoción total queda en 1 ya que *-1+2=1*.
* Finalmente de la tercera a la cuarta la espectacularidad disminuye, la emoción total **disminuye** en 1 porque *3-2=1* y queda en 0 porque *1-1=0*.

Las operaciones para calcular la emoción total iniciando desde la primera vista serían:(1 – 2) + (3 – 1) + (2 – 3) = -1 + 2 -1 = 0Si se inicia en la segunda vista, la *emoción total* se calcula como sigue: (3 – 1) + (2 – 3) + (6 – 2) = 2 – 1 + 4 = 5Revisa con cuidado el cálculo anterior con el ejemplo hasta que entiendas cómo se hizo.Por último si se inicia en la tercera vista, la *emoción total* se calcula: (2 – 3) + (6 – 2) + (2 – 6) = -1 + 4 – 4 = -1No es posible empezar en la cuarta, quinta o sexta vistas ya que sería imposible realizar las 4 vistas del recorrido.**La respuesta del problema es la *emoción total* máxima la cual se obtiene iniciando en la segunda vista y su total es 5.**   |

**Evaluación**

Para un conjunto de casos de prueba con valor de **70** puntos, ***N*** <= 10,000.

Para el resto de los casos de prueba se cumplirá que ***N*** <= 50,000.

**Salsa “La Pikina”**

Tiempo límite por caso: 1 seg

 Memoria límite por caso: 128 MB

Te contrataron para estar a cargo del área de exportaciones de la fábrica de salsa “La Pikina”. Te acaba de llegar un pedido del extranjero. Tu cliente quiere la mayor cantidad de botellas de Salsa la Pikina que puedas enviar, sin embargo debido a regulaciones gubernamentales es necesario que todas las botellas que envíes contengan exactamente la misma cantidad de salsa.

­El sistema de llenado utilizado por tu fábrica tiene un alto margen de error lo que hace imposible tener todas las botellas con la misma cantidad de salsa.

Actualmente tienes ***N*** botellas de salsa en el almacén, cada una tiene una cierta cantidad de salsa. Cuentas también con un total de ***K*** mililitros de salsa, cada uno de los cuales puedes agregar con un máquina a la botella o botellas que desees.

**Problema**

Escribe un programa que decida cómo agregar salsa a las botellas de modo que la cantidad de botellas que puedas enviar sea la máxima posible, es decir, que la cantidad de botellas con exactamente la misma cantidad de salsa sea la máxima posible usando a lo más ***K*** mililitros extras.

Tu programa deberá escribir la cantidad de botellas y la cantidad de salsa que contendrán. En caso de que haya más de una solución con la misma cantidad de botellas deberás elegir aquella en que las botellas tengan la menor cantidad de salsa.

**Entrada**

La primera línea contendrá dos enteros ***N*** y ***K*** respectivamente.

La segunda línea contendrá ***N*** enteros representando la cantidad de mililitros que tiene cada una de las botellas.

**Salida**

Tu programa deberá escribir dos enteros separados por un espacio representando la cantidad máxima de botellas que puedes enviar y los mililitros que contendrá cada una de ellas.

**Restricciones**

**1 <= *N* <= 2,000,000** Número de botellas.

**0 <= *K* <= 1,000,000,000** Cantidad de mililitros.

**0 <= c*i* <= 1,000,000** Mililitros que contiene la ***i-***ésima botella.

**Ejemplo**

|  |  |  |
| --- | --- | --- |
| **Entrada**  | **Salida**  | **Explicación** |
| 5 21 3 3 4 3 | 4 3 | Es posible agregar 2 mililitros a la primera botella con lo cual esta quedaría con 3 mililitros y se tendrían 4 botellas con esa cantidad.Si se quisieran dejar 4 mililitros en todas las botellas sería necesario agregar 3 a la primera botella y 1 a todas las botellas de 3 mililitros, en total 6. Cómo sólo cuentas con 2 mililitros para agregar, esta solución no es posible.  |

**Evaluación**

Para un grupo de casos con valor de 8 puntos, ***N<=*5** y ***K*<=10.**

Para un grupo de casos con valor de 7 puntos, ***N<=*100,000** y ***K*<=1.**

Para un grupo de casos con valor de 14 puntos, ***N<=*1,000.**

Para un grupo de casos con valor de 40 puntos, ***N<=*100,000.**

Para el último grupo de casos con valor de 31 puntos, ***N<=*2,000,000** y ***K*<=1,000,000,000.**

**Ferrocarril “El Chepe”**

Tiempo límite por caso: 1 seg

 Memoria límite por caso: 32 MB

Una de las grandes atracciones turísticas de Chihuahua es el legendario ferrocarril “El Chepe”, en esta ocasión está preparándose para salir en busca de una nueva aventura. Como presidente de la Comisión de Ordenamiento de Maquinaria Interestatal (COMI) estás a cargo de preparar todo para que el viaje sea un éxito.

El Chepe tiene ***N*** vagones, numerados del **1** al ***N***. Los vagones están colocados sobre las vías del ferrocarril, pero desconoces su orden. La COMI te ha encargado ordenar los vagones del ***1*** al ***N*** para garantizar la integridad del ferrocarril.

Para ordenar los vagones tienes a tu disposición una máquina de intercambios. Esta máquina te permite realizar las siguientes operaciones:

* Para un segmento continuo de vagones de tamaño par, la máquina puede intercambiar la mitad izquierda con la mitad derecha del segmento. Ambas mitades mantienen su orden original (observa la imagen de abajo).
* Buscar la posición actual del vagón con número ***v***.
* Saber cuál es el número del vagón en la posición ***p***.

La COMI ha sido muy estricta al aprobar el presupuesto de tu trabajo, por lo que te ha autorizado una cantidad muy limitada de operaciones de cada tipo que puedes realizar para completar el reacomodo.

**Problema**

Dado el número ***N*** de vagones del Chepe, tu tarea es ordenarlos haciendo uso de las operaciones que autorizó la COMI. **Este problema es interactivo**, por lo que no tendrás que leer entrada ni imprimir salida, en vez de eso, tienes que implementar la función AcomodarFerrocarril (**revisa la sección “Implementación”**).

**Implementación**

*Función del evaluador:*EncontrarVagon()

C / C++ int EncontrarVagon(int v);

Pascal function EncontrarVagon(var v: LongInt): LongInt;

*Descripción:* Llama a esta función para saber la posición actual del vagón numero ***v***. Si existe el vagón con número ***v*** la función devolverá un entero indicando su posición, en caso de que pases un número de vagón que no existe, devolverá 0.

*Parámetros:* ***v*** – El número del vagón a encontrar.

*Función del evaluador:*VagonEnPosicion()

C / C++ int VagonEnPosicion(int p);

Pascal function VagonEnPosicion(var p: LongInt): LongInt;

*Descripción:* Llama a esta función para conocer el número del vagón que se encuentra en la posición ***p*** de la vía. En caso de que la posición ***p*** sea menor a **1** o mayor a ***N***, la función devolverá 0.

*Parámetros:* ***p*** – La posición que se quiere consultar.

*Función del evaluador:*IntercambiarVagones()

C / C++ int IntercambiarVagones(int a, int b);

Pascal function IntercambiarVagones(var a, b: LongInt): LongInt;

*Descripción:* Llama a esta función para intercambiar el segmento ***[a, b]*** de vagones. Si ***a*** y ***b*** definen un segmento inválido la función **NO** intercambiará ningún vagón y devolverá 0.Si fue posible realizar el intercambio devolverá 1.

Un segmento es inválido si su tamaño es impar, si ***b < a*** o algún índice es menor a **1** o mayor a ***n***.

*Parámetros:* ***a*** – El índice de inicio del segmento a intercambiar. ***b*** – El índice de término del segmento a intercambiar.

***Tu procedimiento****:* AcomodarFerrocarril()

C / C++ void AcomodarFerrocarril(int n);

Pascal procedure AcomodarFerrocarril(var n: LongInt);

*Descripción:* Esta función recibe el entero ***N*** y debe usar las operaciones anteriores para acomodar el ferrocarril.

*Parámetros:* ***N*** – Un entero indicando el número de vagones del Chepe.

**Ejemplo**

Supón que el Chepe tiene 5 vagones y están acomodados de la siguiente manera: 1, 4, 5, 3, 2. A continuación se muestra cómo se pueden ordenar los vagones del Chepe utilizando las funciones autorizadas por la COMI:

|  |  |  |
| --- | --- | --- |
| **Función llamada** | **Valor devuelto** | **Explicación** |
| AcomodarFerrocarril(5) | - | Esta es la llamada que el evaluador realiza a tu función e inicia tu procedimiento para acomodar. |
| BuscarVagon(1) | 1 | El vagón número 1 se encuentra en la posición 1, por lo tanto este vagón ya se encuentra ordenado. |
| VagonEnPosicion(2) | 4 | El vagón que se encuentra en la posición 2 es el 4. |
| IntercambiarVagones(2,5) | 1 | Se realiza un intercambio de todos los vagones desde la posición 2 hasta la posición 5.Los vagones quedarían de la forma: 1, 3, 2, 4, 5. |
| BuscarVagon(6) | 0 | El vagón 6 no existe porque el Chepe tiene sólo 5 vagones, por lo tanto, la función devuelve 0. |
| BuscarVagon(2) | 3 | El vagón número 2 se encuentra en la posición 3. |
| IntercambiarVagones(3,2) | 0 | No es posible realizar un intercambio cuando el índice de término es menor al índice de inicio. |
| IntercambiarVagones(2,3) | 1 | Después de esta operación los vagones quedarían acomodados de la forma: 1, 2, 3, 4, 5.Una vez que los vagones del tren están ordenados, tu función debe terminar. |

**Evaluador**

El evaluador de prueba recibe del archivo sample.in un entero ***N***, el número de vagones por acomodar, seguido de una línea con ***N*** enteros entre el **1** y el ***N*** describiendo el orden inicial de los vagones del ferrocarril.

Durante la ejecución, el evaluador de prueba imprimirá algunos mensajes para ayudarte a depurar tu solución. Al finalizar la ejecución de tu función, el evaluador de prueba imprimirá la cantidad de veces que llamaste a las funciones EncontrarVagon, VagonEnPosicion e IntercambiarVagones y te dirá si el ferrocarril fue ordenado correctamente.

**Restricciones**

***Subtarea 1 [17 puntos]***

**1 <= *N* <= 200** El número de vagones del Chepe.

Puedes llamar a EncontrarVagon, VagonEnPosicion e IntercambiarVagones a lo más 40,000 veces cada una.

***Subtarea 2 [23 puntos]***

**1 <= *N* <= 2,000** El número de vagones del Chepe.

Puedes llamar a IntercambiarVagones a lo más 50,000 veces.

Puedes llamar a EncontrarVagon y VagonEnPosicion a lo más 5,000 veces cada una.

***Subtarea 3 [25 puntos]***

**1 <= *N* <= 20,000** El número de vagones del Chepe.

Puedes llamar a IntercambiarVagones a lo más 40,000 veces.

Puedes llamar a EncontrarVagon y VagonEnPosicion a lo más 20,000 veces cada una.

***Subtarea 4 [11 puntos]***

**1 <= *N* <= 24,000** El número de vagones del Chepe.

Puedes llamar a IntercambiarVagones a lo más 36,000 veces.

Puedes llamar a EncontrarVagon y VagonEnPosicion a lo más 24,000 veces cada una.

***Subtarea 5 (NO AGRUPADA) [12 puntos]***

**1 <= *N* <= 5,000** El número de vagones del Chepe.

Puedes llamar a IntercambiarVagones a lo más 7,500 veces.

Puedes llamar a VagonEnPosicion a lo más 5,000 veces.

**No puedes hacer ninguna llamada** a EncontrarVagon.

***Subtarea 6 (NO AGRUPADA) [12 puntos]***

**1 <= *N* <= 40,000** El número de vagones del Chepe.

Puedes llamar a IntercambiarVagones a lo más 60,000 veces.

Puedes llamar a VagonEnPosicion a lo más 40,000 veces.

**No puedes hacer ninguna llamada** a EncontrarVagon.

**Evaluación**

Si durante la ejecución de tu función AcomodarFerrocarril excedes el límite de llamadas permitidas para cualquiera de las funciones del evaluador, recibirás un puntaje de 0 puntos. Si al finalizar la ejecución de tu función el ferrocarril no se encuentra ordenado, tu puntaje será de 0 puntos. Recibirás los puntos completos para un caso si el ferrocarril es acomodado correctamente, no excedes el límite de llamadas para ninguna de las funciones del evaluador y tu código no excede el límite de tiempo de ejecución permitido.

**Artesano Rarámuri**

Tiempo límite por caso: 1seg

Memoria límite por caso: 64MB

Un artesano rarámuri desea hacer un tapete conmemorativo de la OMI. Para hacerlo utilizará un gran tapete blanco de tamaño ***N*** x ***N***.

Al artesano le gustan los diseños con cuadros, para este tapete hizo un diseño de cuadrados concéntricos que se alternan de colores negro y blanco. Es importante notar que el **cuadrado exterior siempre será de color negro**. A continuación se muestra una figura ejemplo del diseño del artesano.

Recordarás que el tapete original es blanco, para obtener los cuadros negros, el artesano deberá teñir secciones de la palma con la que se tejió el tapete. Teñir palma es más difícil que teñir tela ya que para **teñir cada sección** de la palma se requiere **distinta cantidad de tinta**.

Por desgracia, el artesano tiene una cantidad ***T*** de tinta y por el momento no puede conseguir más. Debido a esto, es posible que la tinta que tiene **no le alcance para teñir todo el tapete**.

El artesano quiere mantener el diseño y hacer el tapete más grande posible que le alcance con la tinta que tiene (lo que le sobre de tapete será recortado). El artesano sabe **cuánta tinta requiere para teñir cada sección del tapete.**

El artesano te ha pedido tu ayuda para que hagas un programa que dado el tamaño del tapete original, la cantidad de tinta requerida para teñir cada sección y la cantidad total ***T*** de tinta calcule cuál es el tapete más grande que puede hacer respetando su diseño original.

**IMPORTANTE: Recuerda que el cuadro exterior del tapete siempre debe ser de color negro. El cuadro interno será negro o blanco dependiendo del tamaño del tapete.**

**Problema**

Escribe un programa que dado el tamaño del tapete original, la cantidad de tinta necesaria para teñir cada sección y la cantidad ***T*** de tinta que tiene el artesano calcule el tamaño máximo de tapete que puede hacer el artesano respetando su diseño.

**Entrada**

Tu programa debe leer del teclado la siguiente información:

* En la primer línea los números ***N*** y ***T***, el tamaño del tapete original y la cantidad total de tinta que tiene el artesano.
* En las siguientes ***N*** líneas habrá ***N*** números separados por espacio que representan la cantidad necesaria de tinta para teñir cada sección del tapete.

**Salida**

Tu programa deberá escribir a la pantalla un único número entero que representa el largo, en secciones, del cuadro exterior del tapete más grande que pueda hacer el artesano.

**Restricciones**

**1 <= *N* <= 1,000** El tamaño del tapete original.

**1 <= *T* <= 1,000,000,000** Cantidad total de tinta que tiene el artesano.

**1 <= *cij*** **<= 1,000** Cantidad de tinta necesaria para teñir la sección ***i,j***.

**Ejemplo**

|  |  |  |
| --- | --- | --- |
| **Entrada**  | **Salida**  | **Explicación** |
| 8 809 5 3 6 9 1 6 99 7 3 3 1 4 7 56 4 7 7 5 2 5 83 6 6 6 1 3 8 43 9 4 7 5 2 3 79 5 2 9 7 8 7 91 4 5 2 8 5 2 16 2 6 3 8 1 7 5 | 5 | La figura muestra el tapete original y el costo de teñir cada sección del mismo. Recuerda que el artesano desea crear el tapete **más grande** que respete el diseño original. Y de acuerdo al ejemplo, la cantidad total de tinta que tiene es 80.El cuadrado más grande que se puede teñir con la tinta total que tiene el artesano es el que se muestra sombreado en la figura. El largo del cuadro exterior es de 5 secciones y por lo tanto la respuesta es 5. |

**Evaluación**

Para un grupo de casos con valor de 10 puntos, ***N<=*30.**

Para un grupo de casos con valor de 14 puntos, ***N<=*120.**

Para un grupo de casos con valor de 37 puntos, ***N<=*300.**

Para el último grupo de casos con valor de 39 puntos, ***N<=*1,000.**