Problema 1. (3.5 puntos) Dada una serie de números enteros queremos obtener el número de segmentos de la serie de longitud mayor o igual que L que cumplen que cada valor del segmento no difiere del valor siguiente en más de una unidad, y el comienzo de cada uno de los segmentos que lo cumplen. Dado un segmento con un número mayor que L de valores que no difieren en más de una unidad cada uno con el siguiente, se contará únicamente el segmento completo. Por ejemplo, dada la serie 2 2 2 2 7 3 3 con valor $L = 2$ se contarán dos segmentos, el 2 2 2 2 y el 3 3 y se devolverá en un vector las posiciones 0 y 4.

Se pide:

1. (1 punto) Define un predicado `todoLLano` que dada una secuencia indique si sus valores cumplen que cada uno de ellos no difiere del siguiente en más de una unidad.

   Define la precondición del algoritmo.

   Define una postcondición que permita probar que el algoritmo implementado cumple que los valores del vector de salida son el comienzo de segmentos de longitud L que cumplen el predicado `todoLLano`. Por ejemplo si $v_1 \dots v_k$ son los valores calculados por el algoritmo, los segmentos $[v_i \dots v_i + L]$ deben cumplir el predicado `todoLLano`.

2. (1.5 puntos) Implementa un algoritmo que resuelva el problema.

3. (0.5 puntos) Define un invariante del bucle que permita probar que el algoritmo verifica la propiedad pedida en la postcondición y una función cota que permita probar la terminación del algoritmo.

4. (0.5 puntos) Indica el coste del algoritmo implementado y justifícalo.

Entrada

La entrada consta de una serie de casos de prueba. Cada caso de prueba consta de dos líneas. En la primera línea se indica el número de valores de la serie, y el valor de la longitud mínima permitida, $(2 \leq L \leq n)$. En la segunda línea se muestran los valores de la serie $(-1000 \leq v \leq 1000)$.

La entrada acaba con una línea con dos ceros.

Salida

Para cada caso de prueba se escribe en una línea el número de segmentos que cumplen las propiedades pedidas, seguido del comienzo de cada segmento.

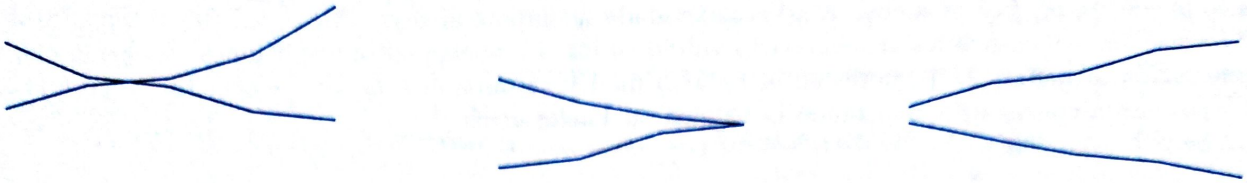
Entrada de ejemplo

```
10 3
5 5 3 4 3 4 7 8 9 3
5 2
5 4 4 5 6
6 3
4 6 8 2 5 9
6 6
3 4 3 2 5 4
0 0
```

Salida de ejemplo

```
2 2 6
1 0
0
0
```

Problema 2 (3 puntos) Tenemos dos series de números enteros, de la misma longitud ($n > 0$), ordenadas, la primera en orden estrictamente creciente y la segunda en orden estrictamente decreciente. Si suponemos que estas series representan dos funciones, se pide determinar las posiciones entre las que se encontraría el punto de corte de dichas funciones. Como se ilustra en los siguientes dibujos:



1. Implementa un algoritmo que resuelva el problema aplicando la técnica de divide y vencerás. El coste debe ser lo más eficiente posible.
2. Indica el coste de la solución obtenida. Debes justificarlo escribiendo y resolviendo la correspondiente recurrencia.

Entrada

La entrada consta de una serie de casos de prueba. Cada caso de prueba consta de tres líneas, en la primera se muestra un valor que representa el número de elementos de las dos series. En la línea siguiente se muestran los valores de la primera serie y en la tercera línea los de la segunda serie.

La entrada termina con una línea con un cero.

Salida

Para cada caso de prueba se escribe en una línea `SI` si existe una posición en que coinciden los valores de las dos series, seguido de la posición en que coincide el valor, o `NO` si no existe dicha posición seguido de las posiciones entre las que se debería encontrar el valor común. Si el punto *virtual* en que se cruzan es anterior al comienzo de las series (como ocurre en el último dibujo) se devolverán las posiciones $-1, 0$. Si el punto en el que se cruzan es posterior al último valor, las posiciones serán $n - 1, n$.

Entrada de ejemplo

```
5
1 3 5 7 9
8 6 4 2 0
3
4 6 8
8 6 4
4
1 2 3 4
4 3 2 1
1
3
5
6
1 2 3 4 5 6
-1 -2 -3 -4 -5 -6
4
-1 1 4 8
1 -1 -3 -10
6
-5 -3 -1 0 4 7
9 6 5 2 1 0
5
1 2 3 4 5
20 19 18 17 16
0
```

Salida de ejemplo

```
NO 1 2
SI 1
NO 1 2
NO 0 1
NO -1 0
NO 0 1
NO 3 4
NO 4 5
```

Problema 3. (3.5 puntos) Cansados de recibir millones de cartas cada año que deben leer primero y buscar después los diferentes juguetes solicitados, Melchor, Gaspar y Baltasar han decidido implantar un sistema más automatizado. Para este año van a hacer una prueba con J juguetes y N niños a los que han solicitado que indiquen su grado de satisfacción (del 1 al 100) con cada uno de los juguetes. Además, se debe tener en cuenta que cada juguete tiene una edad "recomendada", de forma que un niño solo puede recibir juguetes cuya edad recomendada no supere la suya. Ahora los Reyes tienen toda esa información, tienen que hacer un reparto válido de los J juguetes entre los N niños, de forma que cada uno reciba al menos M juguetes en el reparto final y se maximice la satisfacción total. Ayuda a sus Majestades a conseguirlo, utilizando la técnica de *Vuelta atrás*.

Se pide:

1. Definir un *espacio de soluciones* y un *árbol de exploración* adecuados para resolver el problema.
2. Implementar un algoritmo que resuelva este problema de optimización utilizando adecuadamente el esquema de *Vuelta atrás*.

Se valorará la eficiencia de la solución. Deben realizarse el mayor número posible de podas del árbol de exploración, evitando llamadas que no puedan dar lugar a soluciones correctas.

Entrada

Cada caso de prueba consta de una primera línea con el número de juguetes $J > 0$, el número de niños $N > 0$ y el número mínimo de juguetes $M > 0$. Se garantiza que $J \geq N \times M$. A continuación aparece una línea con J valores (entre 0 y 15), que son las edades recomendadas de los juguetes, después otra línea con N valores (entre 0 y 15), que son las edades de los niños, y N líneas más, cada una con J valores (entre 1 y 100) que representan el grado de satisfacción de cada niño con cada juguete.

La entrada termina con una línea con tres ceros (que no se deben procesar).

Salida

Para cada caso de prueba se escribe en una línea el grado máximo de satisfacción que se puede obtener repartiendo los J juguetes entre los N niños, de forma que cada uno reciba al menos M juguetes y la edad recomendada de estos juguetes no supere su edad.

Entrada de ejemplo

```
8 3 2
6 5 0 1 2 9 10 4
6 2 15
5 0 2 2 2 10 10 10
10 10 10 10 10 10 10 10
1 2 3 4 5 6 7 8
0 0 0
```

Salida de ejemplo

```
60
```