Estructuras de Datos y Algoritmos

Grado en Ingeniería del Software y Doble Grado en Informática y Matemáticas

Examen Parcial, 10 de Febrero de 2017.

Normas de realización del examen

1. Debes programar soluciones para cada uno de los tres ejercicios, probarlas y subirlas al juez. El juez se utiliza exclusivamente para la entrega del ejercicio, y no realiza ninguna prueba adicional a los casos del ejemplo. Por lo tanto el veredicto del juez no es significativo y no debe tomarse como que el programa es correcto.

2. Para que los problemas estén presentados deben estar subidos al juez, aunque no estén correctos. La dirección es `http://exacrc/domjudge/team`. Debe escribirse la dirección completa.

3. En el juez te identificarás con el nombre de usuario y contraseña que has recibido al comienzo del examen. El nombre de usuario y contraseña que has estado utilizando durante la evaluación continua **no** son válidos.

4. Escribe tu nombre y apellidos, el laboratorio y el puesto en el que estás y el usuario que vas a utilizar en el juez de examen en la primera línea de cada fichero que subas al juez.

5. Puedes descargar el fichero `http://exacrc/Documentacion_DG_E.zip` que contiene material que puedes utilizar para la realización del examen (plantilla de código fuente y ficheros de texto con los casos de prueba de cada ejercicio del enunciado).

6. Para obtener la máxima puntuación, las soluciones deberán seguir los criterios exigidos a los ejercicios durante el curso (en cuanto a eficiencia, simplicidad, buena programación, comentarios etc.)

7. Tus soluciones serán evaluadas por el profesor independientemente del veredicto del juez automático. Para ello, el profesor tendrá en cuenta exclusivamente el último envio que hayas realizado del ejercicio.

8. Las preguntas del examen que no sean implementación de código se responderán en papel. Debe ponerse el nombre y apellidos del alumno en el papel y entregarlo al finalizar el examen.

Ejercicio 1

(4 puntos) Dado un vector de números enteros positivos, ordenado en orden estrictamente creciente, se pide diseñar un algoritmo eficiente que elimine todos los números impares, dejando sólo los pares, ordenados de forma creciente. El procedimiento modificará el vector de entrada y su tamaño, y no deberá utilizar ningún array auxiliar. Los parámetros por lo tanto serán de entrada/salida.

Se pide:
(1 punto) Especificar el problema.
(1,5 puntos) Diseñar e implementar un algoritmo que resuelva el problema.
(1 punto) Escribir un invariante y una función cota que permitan demostrar la corrección del algoritmo implementado.
(0,5 puntos) Justificar el coste del algoritmo.

Entrada

La entrada que espera el corrector automático consta de una serie de casos de prueba y acabará cuando se introduzca un -1. Cada caso de prueba consta de dos líneas, en la primera se indica el número de elementos del vector. En la segunda se dan los elementos, ordenados en orden estrictamente creciente y separados por el carácter blanco.

Se cumple que el número de elementos del vector es mayor o igual que cero y cada componente del vector es estrictamente mayor que cero.

Salida

Para cada caso de prueba se escribe una línea con los elementos que quedan en el vector después de eliminar los elementos impares dejando un espacio blanco entre cada par de elementos. Si en el vector no queda ningún elemento se deja una línea en blanco.

Entrada de ejemplo

```
5
3 5 6 8 10
4
2 4 5 7
5
1 3 5 7 9
5
2 4 6 8 10
6
3 6 8 9 13 14
-1
```

Salida de ejemplo

```
6 8 10
2 4

2 4 6 8 10
6 8 14
```

Ejercicio 2

(3 puntos) Se tiene un vector con $k > 0$ valores enteros diferentes. Cada valor se encuentra repetido k_i veces, encontrándose consecutivos todos los valores iguales del vector. Se pide encontrar el número de valores distintos.

El problema se debe resolver empleando la técnica de divide y vencerás. Si existen valores repetidos, el vector no debe recorrerse completo, evitando siempre que se pueda el acceso a los elementos repetidos.

Entrada

La entrada que espera el corrector automático consta de una serie de casos de prueba y acabará cuando se introduzca un 0. Cada caso de prueba consta de dos líneas: en la primera se indica el número de elementos del vector y en la segunda los valores separados por un espacio.

Salida

Para cada caso de prueba se escribe el número de valores diferentes del vector en una línea.

Entrada de ejemplo

```
4
3 3 1 1
3
2 2 2
8
6 6 3 3 8 8 1 1
10
3 3 3 6 6 2 2 2 2 2
14
7 7 7 7 7 4 4 4 4 9 9 2 2 2
20
5 5 5 5 5 5 5 5 4 4 4 4 4 4 1 1 1 7 9 2
1
5
6
6 5 5 5 5 2
6
8 7 6 5 4 3
0
```

Salida de ejemplo

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

Ejercicio 3

(3 puntos) Laura quiere construir una torre con piezas de colores. En su juego de construcciones hay piezas azules, rojas y verdes, de cada una de las cuales tiene un determinado número disponible, respectivamente a, r y v. Quiere construir una torre que contenga $n >= 2$ piezas en total cada una encima de la anterior. No le gusta el color verde, asi que nunca coloca dos piezas verdes juntas, ni permite que el número de piezas verdes supere al de piezas azules en ningún momento mientras se va construyendo la torre. Además, como el color rojo es su favorito, las torres que construye siempre tienen en la parte inferior una pieza roja, y en la torre final el número de piezas rojas debe ser mayor que la suma de las piezas azules y verdes.

Implementar un algoritmo que muestre todas las formas posibles que tiene de construir una torre de la altura deseada cumpliendo con las restricciones mencionadas.

Entrada

La entrada que espera el corrector automático consta de una serie de casos de prueba y acabará cuando se introduzca una línea con cuatro ceros. Cada caso de prueba se escribe en una línea y consta de 4 enteros separados por blancos. El primero es mayor que uno y representa la altura de la torre. Los tres siguientes son mayores o iguales que cero y representan los cubos de color azul, rojo y verde respectivamente.

Salida

Para cada caso de prueba se escriben todas las posibles torres, una en cada línea ordenadas por orden lexicográfico y separando cada par de colores por un espacio. Cada caso termina con una línea en blanco. Si no se puede construir la torre con los números de bloques dados se escribirá *SIN SOLUCION*.

Entrada de ejemplo

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

Salida de ejemplo

```
rojo azul rojo rojo
rojo rojo azul rojo
rojo rojo rojo azul
rojo rojo rojo rojo

SIN SOLUCION

rojo azul azul rojo rojo
rojo azul rojo azul rojo
rojo azul rojo rojo azul
rojo azul rojo rojo verde
rojo azul rojo verde rojo
rojo azul verde rojo rojo
rojo rojo azul azul rojo
rojo rojo azul rojo azul
rojo rojo azul rojo verde
rojo rojo azul verde rojo
rojo rojo rojo azul azul
rojo rojo rojo azul verde

rojo rojo
```