Control PRO2 - Turno 1 (Otoño 2017) X17042_es

El sistema jutge.org (de ahora en adelante Jutge) es un entorno de aprendizaje virtual en el que los estudiantes pueden encontrar distintas colecciones de problemas. Dado un problema, un estudiante puede realizar varios envíos con posibles soluciones. Un envío se caracteriza por un conjunto de atributos, tales como la identidad del estudiante que realizó el envío, la fecha y hora en la que se entregó la solución, y el veredicto del Jutge con respecto a los juegos de prueba del problema, que se utilizan para comprobar la corrección de la solución enviada.
Algunos exámenes de asignaturas de programación se realizan y evalúan parcialmente utilizando el Jutge. La calificación automática de los envíos de un estudiante se calcula seleccionando en primer lugar **el mejor de sus envíos** , y multiplicando a continuación el número de juegos de prueba superados por dicho envío por una constante dada (e.g. 2.5 si el problema contiene cuatro juegos de prueba). El mejor envío de un estudiante es el envío de dicho estudiante que supera el máximo número de juegos de prueba. Si el estudiante ha hecho varios envíos que superan el máximo número de juegos de prueba, el mejor envío de dicho estudiante es el último envío que supera el máximo número de juegos de prueba.
En este ejercicio, vamos a construir un programa que lea una secuencia de envíos de soluciones para un problema del Jutge; los almacene en un vector v de envíos; los ordene crecientemente por el DNI del estudiante que realizó el envío, y crecientemente por tiempo de entrega en el caso en que dos envíos hayan sido realizados por el mismo estudiante; y los escriba ordenados de este modo en la pantalla.
La secuencia de envíos viene precedida por el número de estudiantes que pueden realizar envíos, i.e. el número de estudiantes matriculados en el curso del Jutge al que pertenece el problema, y termina con un envío de un estudiante inexistente con DNI igual a 0 (ver el fichero correspondiente a la entrada del ejemplo de este enunciado).
A continuación, el programa recibirá una secuencia de instrucciones, ejecutará las operaciones correspondientes a cada instrucción, y terminará cuando reciba la instrucción "fi".
La instrucción "consultar" requiere leer un entero x correspondiente al DNI del estudiante cuyos envíos se desea consultar. Si no hay ningún envío de dicho estudiante en el vector v, el programa lo indicará mediante un mensaje; en otro caso, escribirá los envíos del estudiante con DNI igual a x que contiene el vector v en orden creciente por tiempo de entrega del envío.
La instrucción "classificar" escribirá el subconjunto de v formado por el mejor envío de cada estudiante ordenado de manera que estén juntos todos los envíos que superan el mismo número de juegos de prueba.
La primera vez que se introduzca la instrucción "classificar" será necesario construir una matriz de clasificación m con los mejores envíos del vector v, para poder escribir el contenido de dicha matriz de clasificación en la pantalla. Se supone que en todo momento el vector v está ordenado crecientemente por el DNI del estudiante que realiza el envío, y que los envíos de un mismo estudiante en el vector v están ordenados a su vez crecientemente por tiempo de entrega en el Jutge.
La matriz de clasificación m contiene un solo envío por estudiante, que es, además, el mejor de los envíos que contiene el vector v de dicho estudiante. Dados dos envíos e_1 y e_2 de un mismo estudiante, e_1 es mejor que e_2 si e_1 supera más juegos de pruebas que e_2, o si e_1 y e_2 superan el mismo número de juegos de pruebas pero e_1 es más reciente que e_2 (i.e. el tiempo de entrega de e_1 es mayor que el de e_2).

Los envíos de la matriz de clasificación m están clasificados, a su vez, por el número de juegos de prueba que superan, de manera que cada fila k de la matriz de clasificación m debe contener únicamente envíos que superen exactamente k juegos de prueba. Dentro de cada fila de la matriz m, los envíos están ordenados crecientemente por el DNI del estudiante que realizó el envío.
En el fichero de salida del ejemplo de este enunciado podéis observar la matriz de clasificación correspondiente a las 29 entregas recibidos. Los envíos que aparecen a continuación de la frase "0 jocs de proves superats" son los envíos de la fila 0 de la matriz de clasificación m. Los envíos que aparecen a continuación de la frase "1 jocs de proves superats" son los envíos de la fila 1 de la matriz de clasificación, y así sucesivamente.
Para implementar este programa hemos construido la clase *Lliurament* (que permite representar un envío –o entrega– de un estudiante) y un módulo funcional *Eines_Vec_Lliu* (que permite realizar distintas operaciones con vectores de objetos de la clase `Lliurament`). Podéis consultar la especificación y la representación del tipo de la clase *Lliurament* en el archivo `Lliurament.hh`, y la especificación del módulo funcional *Eines_Vec_Lliu* en el archivo `Eines_Vec_Lliu.hh`.
Teniendo todo esto en cuenta, debéis implementar eficientemente el método estático y público `millor` de la clase *Lliurament*, que determina si el envío representado por $e1$ es mejor que el envío representado por $e2$.

```
static bool millor(const Lliurament& e1, const Lliurament& e2);
/* Pre: e1 i e2 han estat lliurats pel mateix estudiant.  */
/* Post: Retorna true a algun dels casos següents: 1) e1 ha superat més
 jocs de proves que e2; 2) e1 i e2 han superat el mateix nombre de jocs
 de proves, i el temps de lliurament de e1 és més gran que el temps de
 lliurament de e2. En altres casos, retorna false. */
```

y la acción `classifica` del módulo funcional *Eines_Vec_Lliu*, que construye la matriz de clasificación m descrita anteriormente a partir de un vector v de objetos de la clase `Lliurament`, que en el rango $v[0, \ldots, n\_lliu - 1]$ está ordenado crecientemente por el DNI del estudiante que realizó el envío, y crecientemente por tiempo de entrega en el caso en que dos envíos sean del mismo estudiante.

```
void classifica(int n_lliu, const vector<Lliurament>& v,
     vector<vector<Lliurament> >& m);
  /* Pre: 0 <= n_lliu <= v.size(), v[0 ... n_lliu -1] està ordenat en ordre
  creixent per número de DNI i, en cas d'empat, per temps de lliurament.
  m=M, M.size() = 1 + Lliurament::nombre_jps() i M[j].size()=0 per a tot j */
  /* Post: Per cada x tal que v[0, ..., n_lliu - 1] conté com a mínim un
  lliurament amb DNI = x, m conté el millor lliurament amb DNI = x de
  v[0, ..., n_lliu - 1]. El millor lliurament d'un estudiant és el que
  supera més jocs de proves i, en cas d'empat, el que té el temps de
  lliurament més gran. La matriu m no conté més d'un lliurament amb el
  mateix DNI. A més, els lliuraments de m estan organitzats de la manera
  següent: 1) cada fila k només conté lliuraments que superen exactament
  k jocs de proves; 2) dins d'una fila concreta, els lliuraments estan
  ordenats en ordre creixent per número de DNI. */
```

Debéis entregar un archivo `solucio.cc` con una implementación eficiente del método `millor` de la clase *Lliurament* y de la acción `classifica` del módulo funcional *Eines_Vec_Lliu*. Encontraréis la plantilla del archivo `solucio.cc` dentro del material adicional que os proporcionamos en el apartado *Public files* del problema del Jutge. Esta plantilla se encuentra en el archivo `plantilla.txt`: debéis renombrarlo de manera que se llame `solucio.cc`, completarlo y enviarlo al Jutge.

Vuestro archivo `solucio.cc` no puede contener la implementación de otras operaciones de la clase `Lliurament` ni del módulo funcional `Eines_Vec_Lliu`.

Observación

En el apartado *Public files* del Jutge os proporcionamos material adicional en un fichero .tar. Podéis extraer el contenido de este fichero con la instrucción

```
tar -xvf nom_fitxer.tar
```

Este material adicional contiene los siguientes archivos:

- `plantilla.txt`: es la plantilla del archivo solucio.cc; debéis renombrar este archivo de manera que se llame `solucio.cc`, completarlo y enviarlo al Jutge
- `Lliurament.hh`: la especificación y la representación del tipo de la clase `Lliurament`
- `Lliurament.cc`: la implementación de los métodos de la clase `Lliurament`, excepto la del método estático y público `millor`, que debéis completar en el archivo `solucio.cc`
- `Eines_Vec_Lliu.hh`: la especificación Pre/Post de todas las operaciones del módulo funcional `Eines_Vec_Lliu`
- `Eines_Vec_Lliu.cc`: la implementación de todas las operaciones del módulo funcional `Eines_Vec_Lliu`, excepto la de la acción `classifica`, que debéis completar en el archivo `solucio.cc`
- `pro2.cc`: un programa principal que podéis utilizar para probar los métodos públicos de la clase `Lliurament` y las operaciones del módulo funcional `Eines_Vec_Lliu`
- `llegeixme.txt`: instrucciones para generar el ejecutable del programa `pro2` y probarlo

Valoraremos positivamente que la solución no contenga instrucciones innecesarias (especialmente bucles o llamadas a operaciones costosas), ni objetos (especialmente vectores o matrices) innecesarios, que no haga recorridos cuando debería hacer búsquedas, y que use correctamente las operaciones más adecuadas de la clase `Lliurament` y del módulo funcional `Eines_Vec_Lliu` siempre que sea posible. No se puede usar ninguna estructura de datos que no haya aparecido en las sesiones 1 a 4 de laboratorio. Se permite un uso justificado de la operación `push_back` de la clase `vector`.

Cuando hagáis envíos, el Jutge os indicará cuantos juegos de pruebas supera vuestro programa y de qué tipo (público o privado). El juego de pruebas denominado `público` corresponde a los ficheros `entrada.txt` y `sortida_correcta.txt` del apartado *Public files*.

Ejemplo de entrada

```
13
100 1 1 0 0 2500
102 1 0 0 0 1600
102 1 1 0 0 2400
102 1 1 1 0 4000
105 1 1 1 1 3000
105 1 1 1 0 4000
106 1 1 1 1 3000
120 1 0 1 0 2000
120 1 1 1 0 4000
132 1 1 1 0 2500
132 1 1 1 0 3000
130 1 1 1 1 2600
130 1 1 1 1 2700
135 1 0 0 0 1800
```

```
135 1 0 1 0 2700
100 1 0 0 0 2600
135 1 1 1 0 3600
135 1 0 1 0 4000
136 1 0 1 0 3000
145 1 0 1 0 4000
145 1 1 1 1 4500
152 1 0 1 0 2000
152 1 1 1 0 3000
152 1 1 0 0 4000
156 1 0 0 0 1000
156 1 1 0 0 2000
156 1 1 1 0 3000
156 1 1 1 1 4000
158 1 0 0 0 2000
0
classificar
consultar 156
consultar 103
fi
```

Ejemplo de salida

```
29 LLIURAMENTS REBUTS

DNI 100: [1,1,0,0];  temps: 2500
DNI 100: [1,0,0,0];  temps: 2600
DNI 102: [1,0,0,0];  temps: 1600
DNI 102: [1,1,0,0];  temps: 2400
DNI 102: [1,1,1,0];  temps: 4000
DNI 105: [1,1,1,1];  temps: 3000
DNI 105: [1,1,1,0];  temps: 4000
DNI 106: [1,1,1,1];  temps: 3000
DNI 120: [1,0,1,0];  temps: 2000
DNI 120: [1,1,1,0];  temps: 4000
DNI 130: [1,1,1,1];  temps: 2600
DNI 130: [1,1,1,1];  temps: 2700
DNI 132: [1,1,1,0];  temps: 2500
DNI 132: [1,1,1,0];  temps: 3000
DNI 135: [1,0,0,0];  temps: 1800
DNI 135: [1,0,1,0];  temps: 2700
DNI 135: [1,1,1,0];  temps: 3600
DNI 135: [1,0,1,0];  temps: 4000
DNI 136: [1,0,1,0];  temps: 3000
DNI 145: [1,0,1,0];  temps: 4000
DNI 145: [1,1,1,1];  temps: 4500
DNI 152: [1,0,1,0];  temps: 2000
DNI 152: [1,1,1,0];  temps: 3000
DNI 152: [1,1,0,0];  temps: 4000
DNI 156: [1,0,0,0];  temps: 1000
DNI 156: [1,1,0,0];  temps: 2000
DNI 156: [1,1,1,0];  temps: 3000
DNI 156: [1,1,1,1];  temps: 4000
DNI 158: [1,0,0,0];  temps: 2000

CLASSIFICACIO

0 jocs de proves superats

1 jocs de proves superats
DNI 158: [1,0,0,0];  temps: 2000

2 jocs de proves superats
DNI 100: [1,1,0,0];  temps: 2500
DNI 136: [1,0,1,0];  temps: 3000

3 jocs de proves superats
DNI 102: [1,1,1,0];  temps: 4000
DNI 120: [1,1,1,0];  temps: 4000
DNI 132: [1,1,1,0];  temps: 3000
DNI 135: [1,1,1,0];  temps: 3600
DNI 152: [1,1,1,0];  temps: 3000

4 jocs de proves superats
DNI 105: [1,1,1,1];  temps: 3000
DNI 106: [1,1,1,1];  temps: 3000
DNI 130: [1,1,1,1];  temps: 2700
DNI 145: [1,1,1,1];  temps: 4500
DNI 156: [1,1,1,1];  temps: 4000

LLIURAMENTS DE 156
DNI 156: [1,0,0,0];  temps: 1000
```

```
DNI 156: [1,1,0,0];  temps: 2000
DNI 156: [1,1,1,0];  temps: 3000
DNI 156: [1,1,1,1];  temps: 4000
```

```
NO HI HA LLIURAMENTS DE 103
```

Información del problema

Autor : Professors de PRO2
Traductor : Professors de PRO2
Generación : 2017-10-26 00:20:16