## Tercer Examen de Programación

## Curso 2016-2017

# Concurso ACM

NOTA: Si usted está leyendo este documento sin haber extraído el compactado que se le entregó, ciérrelo ahora, extraiga todos los archivos en el escritorio, y siga trabajando desde ahí. Es un error común trabajar en la solución dentro del compactado, lo cual provoca que los cambios no se guarden. Si usted comete este error y entrega una solución vacía, no tendrá oportunidad de reclamar.

En la facultad de Matemática y Computación de la Universidad de La Habana todos conocen y disfrutan de los concursos ACM-ICPC (*Association for Computing Machinery International Collegiate Programming Contest*). En la competición prima el trabajo en equipo, el análisis de problemas y el desarrollo rápido de software. Unos buscan la perfección en el diseño y codificación de algoritmos, otros tan solo esperan no irse a casa sin algún globo. En esta ocasión eres integrante del equipo organizador de estos concursos y se necesita que contribuyas en un software para el seguimiento y registro de los eventos que tienen lugar en estas competiciones.

En un concurso de este tipo participan N equipos (que para simplificar los vamos a identificar con los números entre 1 y N) donde se enfrentan al desafío de resolver M problemas (también identificados con los números entre 1 y M). Todo el concurso se desarrolla en una plataforma *online* que recepciona los envíos de solución de cada equipo, los evalúa automáticamente y le informa al equipo si su solución fue correcta o no.

De cada envío se registra el instante (número entero) que significará los minutos transcurridos desde que comenzó el concurso al momento en que se registra, el equipo que envió el intento de solución, el problema que intenta resolver y un valor bool que indica si la solución fue correcta o no.

Puede ocurrir que por algún motivo un equipo sea descalificado. Por lo que cualquier consulta que se haga en un instante posterior al de la descalificación no reportará información que involucre a dicho equipo.

Con esta información se puede conocer, entre otras cosas, cuál es la tabla de posiciones en el instante que se le consulte.

Su tarea consiste en hacer una implementación de la *interface* IConcurso que se utilizará para registrar envíos sobre intentos de solución y descalificaciones y permitir diferentes consultas que se explicarán con detalles más adelante.

public interface IConcurso

{

int Penalizacion { get; }

void RegistrarEnvio(int tiempo, int equipo, int problema, bool correcto);

void RegistrarDescalificacion(int tiempo, int equipo);

IEnumerable<int> TablaDePosiciones(int tiempo);

IEnumerable<int> EquiposDescalificados(int tiempo);

IEnumerable<int> ProblemasResueltos(int tiempo, int equipo);

IEnumerable<int> ProblemasConMasSoluciones(int tiempo);

}

El método RegistrarEnvio registra un envío de intento de solución de un equipo a un problema en un instante de tiempo. Si el parámetro correcto tiene valor true significa que la solución fue correcta y si es false significa que la solución fue incorrecta. Se asegura que el parámetro tiempo para todos los métodos siempre será un valor no negativo y los parámetros equipo y problema siempre serán válidos.

El método RegistrarDescalificacion registra la descalificación de un equipo en un instante de tiempo. A partir del instante en que un equipo quede descalificado ya no puede aparecer como resultado de una consulta al método TablaDePosiciones, pero si puede aparecer y debe ser tenido en cuenta en las demás consultas en instantes anteriores.

**Importante:** El parámetro tiempo en los llamados a RegistrarEnvio y RegistrarDescalificacion siempre será secuencial, o sea, su valor en un llamado a RegistrarEnvio o RegistrarDescalificacion siempre será mayor o igual que su valor en cualquier llamado anterior a estos métodos.

El método TablaDePosiciones devuelve los equipos ordenados descendentemente por la cantidad de problemas resueltos. A igual cantidad de problemas resueltos ocupará una mejor posición el que lo haya logrado en menos tiempo.

Existe una penalización por los intentos infructuosos de solución. Cuando se registre una solución como correcta se le sumará al tiempo en que se registra un valor Penalizacion por cada intento fallido de solución que se haya intentado previamente para ese problema.

Note que Penalizacion es una propiedad que en sus pruebas usted puede darle el valor que desee pero que una vez inicializado no debe usted cambiarlo porque los resultados serían disparatados. En las pruebas que haremos de su solución le daremos un único valor inicial.

De modo que para calcular el tiempo consumido por un equipo en sus problemas ya resueltos:

1. Se suman todos los instantes de tiempo de los registros de problemas solucionados correctamente por ese equipo.
2. La suma anterior debe estar incrementada en Penalizacion \* *k* donde *k* es la cantidad de intentos infructuosos previos de los problemas ya resueltos.

Por ejemplo, si el valor de Penalizacion fuese 5 entonces para un equipo que ha hecho los siguientes envíos:

|  |  |  |
| --- | --- | --- |
| **Tiempo (min)** | **Problema** | **Resultado** |
| 1 | 4 | INCORRECTO |
| 2 | 4 | CORRECTO |
| 5 | 2 | INCORRECTO |
| 10 | 3 | CORRECTO |
| 12 | 2 | CORRECTO |
| 14 | 1 | INCORRECTO |
| 20 | 1 | INCORRECTO |

El tiempo consumido en el minuto 11 de sus problemas resueltos es:

1. Hasta el minuto 11 tiene resuelto los problemas 4 y 3. Registrados en los minutos 2 y 10, de modo que 2 + 10 = 12
2. Como tiene un intento fallido del problema 4 antes de resolverlo (minuto 2) entonces el tiempo total consumido de sus soluciones correctas es 12 + 5\*1 **= 17**

Si se quisiera calcular para este mismo equipo en el minuto 15 sería:

1. Hasta el minuto 15 tiene resuelto los problemas 4 (minuto 2), 3 (minuto 10) y 2 (minuto 12). Entonces 2 + 10 + 12 = 24
2. Como tuvo un intento fallido para el problema 4 (en el minuto 1) y un intento fallido para el problema 2 (en el minuto 5) entonces el tiempo total consumido para sus problemas resueltos es 24 + 5\*2 **= 34**

Solo se requiere que el resultado del método TablaDePosiciones esté ordenado de forma **no creciente** en cuanto a la cantidad de problemas resueltos y en caso de empate ordenar de forma **no decreciente** según este criterio de tiempo más penalización. Si después de aplicar estos criterios aún dos equipos siguen empatados, se pueden devolver en cualquier orden, pues a los efectos de la tabla estarían en la misma posición.

El método EquiposDescalificados devuelve el conjunto de equipos que han sido descalificados hasta el instante de tiempo que se le consulta. Note que esta consulta puede ser retrospectiva, es decir podemos consultar por los descalificados en el instante 100 y ya se haya registrado una descalificación en el instante 150, tal descalificación no saldrá entonces en la respuesta.

El método ProblemasResueltos devuelve el conjunto de problemas resueltos por el equipo especificado hasta ese instante de tiempo.

El método ProblemasConMasSoluciones devuelve el conjunto de problemas ordenado de forma **no creciente** por la cantidad de equipos que los han solucionado hasta el instante de tiempo especificado. En caso de que dos problemas hayan sido resueltos por la misma cantidad de equipos se pueden ubicar en cualquier orden.

**Importante:** Es preciso tener en cuenta que se puede consultar a la estructura con cualquier instante de tiempo y esta debe responder correctamente según el estado del concurso hasta dicho momento (incluído el tiempo pasado como parámetro).

Usted debe haber recibido junto a este documento una solución de Visual Studio con dos proyectos: una biblioteca de clases (*Class Library*) y una aplicación de consola (*Console Application*). Usted debe completar la implementación de la clase Concurso en el namespace *Weboo.Examen* que ya implementa la *interface* IConcurso. Tenga en cuenta que esta clase tiene un constructor que recibe como argumentos la cantidad de equipos, la cantidad de problemas y la penalización que se empleará en el concurso al cual se le va a dar seguimiento.

public class Concurso : IConcurso

{

public Concurso(int equipos, int problemas, int penalizacion)

{

throw new **NotImplementedException**();

}

public int Penalizacion

{

get { throw new **NotImplementedException**(); }

}

public void RegistrarEnvio(int tiempo, int equipo, int problema, bool correcto)

{

throw new **NotImplementedException**();

}

public void RegistrarDescalificacion(int tiempo, int equipo)

{

throw new **NotImplementedException**();

}

public IEnumerable<int> TablaDePosiciones(int tiempo)

{

throw new **NotImplementedException**();

}

public IEnumerable<int> EquiposDescalificados(int tiempo)

{

throw new **NotImplementedException**();

}

public IEnumerable<int> ProblemasResueltos(int tiempo, int equipo)

{

throw new **NotImplementedException**();

}

public IEnumerable<int> ProblemasOrdenadosPorCantidadDeSoluciones(int tiempo)

{

throw new **NotImplementedException**();

}

}

NOTA: Todo el código de la solución debe estar en este proyecto (biblioteca de clases), pues es el único código que será evaluado. Usted puede adicionar todo el código que considere necesario, pero no puede cambiar los nombres del namespace, clase o método mostrados. De lo contrario, el probador automático fallará. En particular, es imprescindible que usted no cambie los parámetros del constructor de la clase Concurso, ni su orden. Por supuesto, usted puede (y debe) adicionar todo el código que necesite al cuerpo del constructor para inicializar sus estructuras de datos.

# Ejemplos

//Se crea un nuevo concurso con 8 participantes,

//5 problemas y una penalización de 20 min

Concurso copaUH = new Concurso(8, 5, 20);

//El equipo 1 hace un envío incorrecto al problema 5 en el minuto 3

copaUH.RegistrarEnvio(3, 1, 5, false);

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Equipo** | **Total** | **Tiempo** | **1** | **2** | **3** | **4** | **5** |
| 1 | 1 | 0 | 0 |  |  |  |  | (**-1**) |
| 1 | 2 | 0 | 0 |  |  |  |  |  |
| 1 | 3 | 0 | 0 |  |  |  |  |  |
| 1 | 4 | 0 | 0 |  |  |  |  |  |
| 1 | 5 | 0 | 0 |  |  |  |  |  |
| 1 | 6 | 0 | 0 |  |  |  |  |  |
| 1 | 7 | 0 | 0 |  |  |  |  |  |
| 1 | 8 | 0 | 0 |  |  |  |  |  |

//El equipo 2 hace un envío correcto al problema 5 en el minuto 4

copaUH.RegistrarEnvio(4, 2, 5, true);

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Equipo** | **Total** | **Tiempo** | **1** | **2** | **3** | **4** | **5** |
| 1 | 2 | 1 | 4 |  |  |  |  | **+** |
| 2 | 1 | 0 | 0 |  |  |  |  | (**-1**) |
| 2 | 3 | 0 | 0 |  |  |  |  |  |
| 2 | 4 | 0 | 0 |  |  |  |  |  |
| 2 | 5 | 0 | 0 |  |  |  |  |  |
| 2 | 6 | 0 | 0 |  |  |  |  |  |
| 2 | 7 | 0 | 0 |  |  |  |  |  |
| 2 | 8 | 0 | 0 |  |  |  |  |  |

//El equipo 1 hace un envío correcto al problema 5 en el minuto 5

copaUH.RegistrarEnvio(5, 1, 5, true);

//El equipo 3 hace un envío correcto al problema 5 en el minuto 6

copaUH.RegistrarEnvio(6, 3, 5, true);

//El equipo 4 hace un envío correcto al problema 5 en el minuto 6

copaUH.RegistrarEnvio(6, 4, 5, true);

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Equipo** | **Total** | **Tiempo** | **1** | **2** | **3** | **4** | **5** |
| 1 | 2 | 1 | 4 |  |  |  |  | **+** |
| 2 | 3 | 1 | 6 |  |  |  |  | **+** |
| 2 | 4 | 1 | 6 |  |  |  |  | **+** |
| 3 | 1 | 1 | 25 |  |  |  |  | **+**(**-1**) |
| 4 | 7 | 0 | 0 |  |  |  |  |  |
| 4 | 8 | 0 | 0 |  |  |  |  |  |
| 4 | 5 | 0 | 0 |  |  |  |  |  |
| 4 | 6 | 0 | 0 |  |  |  |  |  |

//El equipo 5 hace un envío correcto al problema 5 en el minuto 10

copaUH.RegistrarEnvio(10, 5, 5, true);

//El equipo 2 hace un envío incorrecto al problema 3 en el minuto 15

copaUH.RegistrarEnvio(15, 2, 3, false);

//El equipo 2 hace un envío correcto al problema 3 en el minuto 17

copaUH.RegistrarEnvio(17, 2, 3, true);

//El equipo 5 hace un envío correcto al problema 3 en el minuto 20

copaUH.RegistrarEnvio(20, 5, 3, true);

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **#** | **Equipo** | **Total** | **Tiempo** | **1** | **2** | **3** | **4** | **5** |
| 1 | 5 | 2 | 30 |  |  | **+** |  | **+** |
| 2 | 2 | 2 | 41 |  |  | **+**(**-1**) |  | **+** |
| 3 | 4 | 1 | 6 |  |  |  |  | **+** |
| 3 | 3 | 1 | 6 |  |  |  |  | **+** |
| 4 | 1 | 1 | 25 |  |  |  |  | **+**(**-1**) |
| 5 | 8 | 0 | 0 |  |  |  |  |  |
| 5 | 7 | 0 | 0 |  |  |  |  |  |
| 5 | 6 | 0 | 0 |  |  |  |  |  |

//Consultas a la tabla de posiciones en los minutos 2, 7 y 30

int[] tablaMin2 = copaUH.TablaDePosiciones(2).ToArray(); //{1, 2, 3, 4, 5, 6, 7, 8}

int[] tablaMin7 = copaUH.TablaDePosiciones(7).ToArray(); //{2, 3, 4, 1, 7, 8, 5, 6}

int[] tablaMin30 = copaUH.TablaDePosiciones(30).ToArray(); //{5, 2, 4, 3, 1, 8, 7, 6}

//No hay equipos descalificados hasta el momento

int[] descMin30 = copaUH.EquiposDescalificados(30).ToArray(); //{ }

//Son descalificados los equipos 4 y 7 en los minutos 33 y 40 respectivamente

copaUH.RegistrarDescalificacion(33, 4);

copaUH.RegistrarDescalificacion(40, 7);

//Notar el cambio antes y después de decalificar al equipo 4

int[] tablaMin32 = copaUH.TablaDePosiciones(32).ToArray(); //{5, 2, 4, 3, 1, 8, 7, 6}

int[] tablaMin34 = copaUH.TablaDePosiciones(34).ToArray(); //{5, 2, 3, 1, 8, 7, 6}

//... y después de descalificar al equipo 7

int[] tablaMin41 = copaUH.TablaDePosiciones(41).ToArray(); //{5, 2, 3, 1, 8, 6}

//Equipos descalificados en los minutos 34 y 41

int[] descMin34 = copaUH.EquiposDescalificados(34).ToArray(); //{4}

int[] descMin41 = copaUH.EquiposDescalificados(41).ToArray(); //{4, 7}

//Problemas resueltos por los equipos 1, 5 y 8 hasta el minuto 16

int[] probEq1Min16 = copaUH.ProblemasResueltos(16, 1).ToArray(); //{5}

int[] probEq5Min16 = copaUH.ProblemasResueltos(16, 5).ToArray(); //{5}

int[] probEq8Min16 = copaUH.ProblemasResueltos(16, 8).ToArray(); //{}

//Problemas resueltos por los equipos 1, 5 y 8 hasta el minuto 45

int[] probEq1Min45 = copaUH.ProblemasResueltos(45, 1).ToArray(); //{5}

int[] probEq5Min45 = copaUH.ProblemasResueltos(45, 5).ToArray(); //{3, 5}

int[] probEq8Min45 = copaUH.ProblemasResueltos(45, 8).ToArray(); //{}

|  |  |  |  |
| --- | --- | --- | --- |
| **Equipo** | **Resueltos Minuto 16** | **Resueltos Minuto 18** | **Resueltos Minuto 45** |
| 1 | 5 | 5 | 5 |
| 2 | 5 | 3, 5 | 3, 5 |
| 3 | 5 | 5 | 5 |
| 4 | 5 | 5 | 5 |
| 5 | 5 | 5 | 3, 5 |
| 6 |  |  |  |
| 7 |  |  |  |
| 8 |  |  |  |

//Problemas con más soluciones en los minutos 0, 7 y 18

int[] probSolMin0 = copaUH.ProblemasConMasSoluciones(0).ToArray(); //{1, 2, 3, 4, 5}

int[] probSolMin7 = copaUH.ProblemasConMasSoluciones(7).ToArray(); //{5, 1, 2, 3, 4}

int[] probSolMin18 = copaUH.ProblemasConMasSoluciones(18).ToArray(); //{5, 3, 1, 2, 4}

NOTA: Los casos de prueba que aparecen en este proyecto son solamente de ejemplo. Que usted obtenga resultados correctos con estos casos no es garantía de que su solución sea correcta y de buenos resultados con otros ejemplos. De modo que usted debe probar con todos los casos que considere convenientes para comprobar la validez de su implementación.