Fábrica de Yeso DAM

Tenemos una fábrica de yesos que produce determinados sacos al día.

Un **saco** tiene un **número de lote**, un **código identificativo**, un **peso** que ronda aleatoriamente según el proceso de producción entre 25 y 50 kilogramos ambos inclusive y una **fecha de producción** en formato dd-mm-aaaa. Además, tenemos **tres tipos de categorías de sacos**. Los sacos normales, que se fabrican el 60% de las veces con el peso indicado. Los sacos súper, que se fabrican el 25% de las veces con un incremento del 20% en el intervalo de peso, y los sacos extra que se fabrican el 15% de las veces con un intervalo del 25% de incremento de peso. Tarda en fabricar el saco un tiempo de igual al peso.

En una jornada de producción, **fabricador**, fabrica 300 sacos como los indicados y se guarda este **manifiesto** en un fichero de tipo texto llamado producción.txt con el siguiente formato: código de saco; numero de lote; peso; fecha de producción; categoría.

Finalmente, dicho cargamento de sacos con su manifiesto llegará al **empaquetador**. Se leerá el manifiesto y cargará los sacos organizándolos en orden de peso usando un TDA árbol. Asignándole el número de lote a cada saco de 10 en 10. Esperando entre cada lote un tiempo que es la suma de los sacos partido 10. Una vez que le hemos asignado el número del lote al saco, y los almacenamos en un XML que deberás crear de acuerdo con las especificaciones indicadas. Por lo tantos tendrás una jerarquía o listas de lotes, cada una de ellas compuestas de sacos.

Una vez terminado el proceso de producción y empaquetado obtendremos un informe en formato Markdown donde mostremos por cada lote: el número del lote, el número de sacos que incluye, el peso total del lote, el peso medio, el peso máximo y el peso mínimo de ese lote.

Resolver intentando hacer uso de la concurrencia lo máximo posible de manera automática.

Jamones para “probar” **(y también para aprobar)**

Se dice y se rumorea que uno de los principales “secretos” para aprobar PSP es la entrega de un jamón del Valle de los Pedroches al profesor, eso es un decir, lo de verdad importante es resolver cómo se realizar el proceso de gestión y de obtención de este importante “secreto”.

Tenemos varias granjas de proveedores de cerdo ibérico, un secadero, y un mensajero. Nuestro secadero, sólo puede admitir un máximo de diez jamones.

Nuestras granjas, dos en total, obtienen el jamón fresco en un intervalo de un segundo. Una vez listo el jamón freso lo llevan al secadero si hay espacio. Producirán 30 jamones en total cada granja. De la granja sabemos que tiene un nombre: “granja y su número”. Además, sabemos que la granja 2 siempre es más eficiente obteniendo jamones y la 1 la menos eficiente. Cada jamon tendrá un id y un peso entre 6 y 9 kilogramos y el identificador de la granja.

Por otro lado, tenemos el mensajero que sacará tandas de tres en tres jamones cada tres segundos si es posible y hay existencias en el secadero, agrupándolos en un lote para llevarlos a la tienda. Una vez sacado imprime el lote con su identificador.

El sistema debe terminar correctamente una vez se hayan producido todos los jamones y se han distribuido por lotes cada uno de ellos. Se debe mostrar información correcta y válida por pantalla de lo que va sucediendo.

Debemos simular el funcionamiento de nuestro sistema mediante hilos, y los mecanismos de sincronización y comunicación adecuados. Se valorará el diseño, el correcto uso de los métodos adecuados, la prevención de errores. Comentar el código es obligatorio.

Se debe entregar el proyecto y una captura de su correcto funcionamiento.

De esta “sutil” manera creo que algún jamón llegará al profesor, aunque sea por la pantalla de la consola.
Al final, el profesor tenia razón. Aprobar es cuestión de jamones.

“*Más vale tener cerdos que dan jamones y no rompen los corazones.”*
Refranero popular

Fabrica de Aceite

Nos han contratado para manejar la cadena de montaje de una fabrica de aceite. En nuestra fábrica tenemos lo siguiente:

Cada segundo llega un cargamento veinte kilos de aceituna, de esas aceitunas se produce un litro de aceite por cada cinco kilos y tardamos tres segundos en producirlo y se almacena en el deposito.

Embotellamos dichos litros de cinco en cinco desde el deposito, por lo que una vez que tenemos los litros necesarios creamos y llenamos la botella en dos segundos insertando la fecha de caducidad que será la fecha de hoy más un año, además tendrá un identificador único por botella y se dejan en el almacén
Finalmente, el empaquetador, empaqueta las botellas, si las hay, en cajas de diez en diez, es decir, una caja, diez botellas. Además por cada caja tendremos su código y la fecha de fabricación completa, incluida hora, minutos y segundos. Tardamos cinco segundos en hacer todo esto.

Una vez esté lista la caja el empaquetador rellena el manifiesto.

El manifiesto será un fichero llamado manifiesto.txt que tiene los datos de la caja y de cada botella de aceite que contiene. Este manifiesto deberá mostrar todo el proceso de producción completo.

Debemos ejecutar el programa durante cinco minutos. No olvides que puedes jugar con los milisegundos para no esperar tanto en cada caso.
Deberás abrir el manifiesto una vez terminada la ejecución.

Tragabolas

Hace mucho tiempo, en una galaxia mu lejana Bueno, no tan lejana; mis padres me regalaron un Tragabolas ya que no me daban Nocilla, así que entre bocata y bocata de jamón echaba una partida con mis vecinos.

Explicaré para vosotros jóvenes imberbes y damiselas inocentes que este "juegaco" consistía en que cuatro jugadores luchaban por una bola que se lanzaba al centro del tablero, accionando una palanca que lanzaba la boca de un hipopótamo que intentaba trincar cada bola.

En cada ronda, se cede por turnos el lanzamiento de la bola (sólo podrá lanzarse una al tablero). El lanzamiento de la bola hará que esta se deposite en una posición aleatoria del tablero.

Una vez en el ruedo la bola es tratada de engullir por cada uno de los cuatro hipopótamos. Pero solo uno puede acceder al tablero a la vez, pero teniendo en cuenta que un hipopótamo tendrá el 33% de éxito de "pegarle el bocado" al tablero. Es decir, a veces de los nervios y el sudor se nos escapaba la palanca, y solo le dábamos una de tres veces.

Entre los que han conseguido darle a la palanca, solo uno llegará a la bola. Cuando la bola es zampada, que solo ocurre con el 20% de probabilidad, otro jugador (el que le toque) accede al centro del tablero y lanza la bola, recordar que el turno es circular.

Debemos tener en cuenta el número de intentos por hipopótamo que ha habido para comerse la bola en cada ronda.

Cada jugador pondrá en juego un total de 5 bolas. Diremos algo como "Ronda X, lanza la bola Jugador Y".

Después se acaba el juego y el hilo principal mostrará cuantas bolas ha comido cada hipopótamo (es decir tendremos un total de 20 partidas, cinco bolas por los cuatro jugadores). Se deberá mostrar un resumen tanto por ronda, como por partida completa.

Por ronda diremos algo así como el "Ronda X, el hipopótamo Y se ha comido la bola en el intento Z".

Finalmente mostraremos la clasificación de la partida de cada hipopótamo ordenado de quien haya comido más bolas al que haya comido menos bolas.
Menudo juegazo Y no las PlaysStation esas que os gastáis vosotros. He estado jugando con esto hasta los 30 y porque me lo han escondido que si no... Bueno eso es otra historia.

Carreras de camellos

Simula la carrera de camellos de las atracciones de feria.
Os recuerdo que dicha atracción consiste en lanzar bolas en un panel que tiene agujeros numerados. Según donde colemos la bola el camello que tenemos asignado avanzará entre 1 y 20 posiciones.

Los avances del 1 al 3 saldrán al 40 %, del 4 al 7 al 30 %, 8 y 9 al 20 % y 10 al 10%.

Cada camello será un hilo que tendrá un nombre y un dorsal asignado.

Demos mostrar en todo momento el resultado de la partida y el podium.
Se recomienda hacerlo con interfaz gráfica. El recorrido de cada camello será un jProgressBar.

Además habrá dos botones, uno para parar la partida y otro para comenzarla. A cada camello se le puede poner un nombre.