PRÁCTICA 2 ÁRBOLES (ABB)

Se trata de realizar un diseño de un programa para la gestión de un cine que, haciendo uso de ABB permita hacer una simulación de la ocupación de las salas. Los datos se generarán de manera aleatoria en base a las especificaciones descritas en este enunciado. El programa deberá estar desarrollado en Python (Practica1_NombreApellidos.py)

Cada persona asociada al cine y que podrá comprar entradas tendrá una tarjeta de fidelización que almacenará inicialmente:

1. **ID de tarjeta**: número de 6 dígitos p.e.: 123456, 654321, 111111, etc.
2. **Nombre**: string de 10 letras p.e.: “abcdefghij”
3. **Edad**: número entre 0 y 99
4. **Género**: string {femenino, masculino, otro}
5. **Entrada** (booleano)

Habrá que crear un ABB para guardar los datos de cada sala del cine. **La clave de ordenación será el número de tarjeta**. Inicialmente, las 3 salas (ABB) están vacías y el interfaz de la aplicación debe permitir al usuario, de forma dinámica, las siguientes operaciones (no se exige que el interfaz sea en modo gráfico):

1. Insertar un miembro en un ABB: 1,2,3
2. Buscar cualquier espectador en un ABB: 1, 2, 3
3. Imprimir lista de espectadores en cualquier ABB indicado recorridos de cualquier forma: anchura, profundidad (PRE, IN, POST ORDEN).
4. Borrar un espectador, cuya tarjeta haya sido introducida por teclado, si existe en el correspondiente ABB.
5. Introducir un número y crear dicha cantidad de espectadores al azar, guardando en el correspondiente ABB estos espectadores: 1,2,3
6. Introducir un número y generar al azar dicha cantidad de espectadores, que se borrarán, si existen, en los correspondientes ABB.
7. Iniciar la simulación.
8. Salir de la aplicación.

Todas las operaciones deben devolver el estado inicial y final de los ABB afectados tras la ejecución de los comandos

SIMULACIÓN

Una vez que se pulse la opción (7) Iniciar la simulación, dejaremos de tener control sobre la aplicación y ésta simulará el comportamiento del sistema durante 60s. Con esta opción de simulación se verá cómo van evolucionando los diferentes ABB, mostrando las entradas y salidas de espectadores a las salas (ABB).

Cada 5 segundos de ejecución llegará un espectador a un ABB al azar. Sus datos se generarán de manera aleatoria y se guardarán en el correspondiente ABB.

Cada 5 segundos de ejecución saldrá un espectactor al azar de uno de los ABB. Su atributo **entrada** será positivo (1, True) o negativo (0, False). Si el valor de este atributo es positivo no se hará nada, pero si el atributo es negativo, este espectador se eliminará del árbol correspondiente.

Se trata de hacer una simulación donde se vea cómo van evolucionando los ABB de las salas de un cine. Habrá que mostrar el estado en el que se encuentran los 3 ABB y los eventos que van ocurriendo, al igual que en la práctica anterior. En este caso, se mostrarán eventos únicamente para las entradas y salidas de espectadores en las salas (ABB).

Durante la simulación habrá que mostrar los ABB en modo **inorden**.

ABB1: 234567

ABB2: 987654

ABB3: 101010, 222222

EVENTO: Se borra espectador 555555 del ABB3.

ABB1: 234567

ABB2: 145685 987654

ABB3: 101010 222222

EVENTO: Se introduce espectador 145685 en el ABB2.

ENTREGA

Se deberá entregar la aplicación funcionando correctamente, junto con una documentación sencilla explicando la definición de la clase ABB realizada en Python y los problemas detectados. La fecha máxima de entrega es la establecida en la tarea disponible en el campus virtual para la **práctica 2**.