Práctica Estructura de Datos

Con esta práctica se pretende simular el funcionamiento de una web especializada en la venta de entradas para conciertos. Los usuarios de dicha web pueden ser de dos tipos: clientes registrados (pagan una cuota de suscripción y disfrutan de determinadas ventajas en la compra de entradas) y clientes no registrados. Para cada cliente se tendrá DNI, hora de inicio de compra, si es registrado o no, tiempo que tarda el sistema en procesar su compra (aleatorio entre 1 y 10 minutos) y un identificador de entrada.

Los clientes no registrados acceden a la venta a partir de las 02:00 horas de la fecha indicada por la organización del concierto mientras que los clientes registrados pueden acceder a la compra de entradas con 2 horas de antelación a dicha hora (00:00).

El sistema creará dos colas de solicitudes en espera, una para cada tipo de cliente, a las que irán llegando los clientes que acceden a la web para comprar las entradas. A continuación, se describe el funcionamiento de dichas colas

A. Los usuarios registrados se sitúan en la cola 1 (cola VIP) y van siendo atendidos en orden de llegada en la cola de atención de un único servidor.
B. Los usuarios no registrados esperan en la cola 2 y van siendo atendidos, por orden de llegada, en 4 servidores diferentes. En cada uno de los servidores se utiliza una cola de atención a la que van llegando las solicitudes de la siguiente forma:
    a. Las solicitudes se sitúan en la cola de atención del primer servidor libre. Las solicitudes saldrán del servidor una vez finalizado el proceso de compra.
    b. En caso de que todos los servidores estén ocupados se situará en la cola de atención del servidor con menos solicitudes.

Todos los usuarios, una vez comprada su entrada, almacenan en una única *lista enlazada* ordenada por identificador[1] de entrada. La lista deberá estar ordenada en todo momento.

Finalmente, de esta *lista enlazada* se creará el ABB ordenado por DNI[2] para que los clientes puedan buscar y descargar rápidamente su entrada cuando quieran.

Opciones del programa:

A. Generar solicitudes de entradas de los clientes VIP. Las solicitudes de los clientes se van generando en orden de llegada, es decir, el tiempo de llegada de cada cliente se calcula a partir del anterior con una diferencia de tiempo de llegada aleatoria entre 0 y 5 minutos. El primer cliente llegará a las 00:00 y el siguiente entre las 00:00 y las 00:05.
B. Mostrar la cola de solicitudes en espera de clientes VIP.
C. Borrar la cola de solicitudes en espera de clientes VIP.
D. Generar solicitudes de entradas de los clientes no registrados. Las solicitudes de los clientes se van generando en orden de llegada, es decir, el tiempo de llegada de cada cliente se calcula a partir del anterior con una diferencia de tiempo de llegada aleatoria entre 0 y 5 minutos. El primer cliente llegará a las 02:00 y el siguiente entre las 02:00 y las 02:05.
E. Mostrar la cola de solicitudes en espera de usuarios no registrados.
F. Borrar la cola de solicitudes en espera de usuarios no registrados.

[1] EL identificar único de la entrada consistirá en cuatro letras generadas aleatoriamente y será asignado a cada entrada antes de insertarla en *la lista de entradas vendidas*.
[2] Debe generarse aleatoriamente un número válido y calcular la letra del mismo.

G. Simular el proceso de compra de entradas de los clientes VIP, en función del tiempo de llegada. A la hora de llegada del cliente se procesará su entrada en la cola de atención del servidor y al finalizar la compra se almacenarán en la *lista de entradas vendidas.* Durante la simulación deberán mostrarse en todo momento la cola de espera, el servidor y la lista enlazada de entradas vendidas.
H. Simular el proceso de compra de entradas de los clientes no registrados, en función del tiempo de llegada. A la hora de llegada del cliente se situará la solicitud en la cola del servidor correspondiente, según se ha explicado anteriormente. Al finalizar la compra se almacenarán en la *lista de entradas vendidas.* Durante la simulación deberán mostrarse en todo momento la cola de espera, los servidores y la lista enlazada de entradas vendidas.
I. Mostrar la lista de entradas vendidas.
J. Reiniciar el programa.

K. Crear un árbol binario de búsqueda ordenado por DNI.
L. Buscar un cliente en el ABB, dado su DNI, y ver sus datos.
M. Mostrar el ABB en preorden.
N. Mostrar el ABB en postorden.
O. Mostrar el ABB en inorden.
P. Dibujar el ABB (opcional).

S. Salir del programa.

main.cpp

```
#include <ctime>
#include <iostream>
#include <iomanip>
#include "windows.h"
using namespace std;

int main(int argc, char** argv)
{
    char opcion;
    ShowWindow(GetConsoleWindow(), SW_MAXIMIZE);
    srand(time(NULL));

    do {

        HANDLE hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
        SetConsoleTextAttribute(hConsole, 74);
        cout << "\n\t--------------------------------------------------------------------------\n";
        cout << "\t                              MENU                                        \n";
        cout << "\t--------------------------------------------------------------------------\n";
        SetConsoleTextAttribute(hConsole, 47);
        cout << "\tNumero de clientes VIP: " << setw(2) << 0
             << " Numero de clientes no registrados:  " << setw(2) << 0 << endl
             << endl;
        SetConsoleTextAttribute(hConsole, 7);

        cout << "\tA. Generar solicitudes de entradas de los clientes VIP.\n";
        cout << "\tB. Mostrar la cola de solicitudes en espera de clientes VIP.\n";
        cout << "\tC. Borrar la cola de solicitudes en espera de clientes VIP.\n";
        cout << "\tD. Generar solicitudes de entradas de los clientes no registrados.\n";
        cout << "\tE. Mostrar la cola de solicitudes en espera de usuarios no registrados.\n";
        cout << "\tF. Borrar la cola de solicitudes en espera de usuarios no registrados.\n";
        cout << "\tG. Simular el proceso de compra de entradas de los clientes VIP.\n";
        cout << "\tH. Simular el proceso de compra de entradas de los clientes no registrados.\n";
        cout << "\tI. Mostrar la lista de entradas vendidas.\n";
        cout << "\tJ. Reiniciar el programa.\n\n";

        cout << "\tK. Crear un arbol binario de busqueda ordenado por DNI.\n";
        cout << "\tL. Buscar un cliente en el ABB, dado su DNI, y ver sus datos. \n";
        cout << "\tM. Mostrar el ABB en preorden.\n";
        cout << "\tN. Mostrar el ABB en postorden.\n";
        cout << "\tO. Mostrar el ABB en inorden.\n";
        cout << "\tP. Dibujar el ABB.\n\n";

        cout << "\tS.Salir del programa.\n\n";
        cout << "\tIndique la opcion deseada: ";
        cin >> opcion;
        opcion = toupper(opcion);
        system("cls");

        switch(opcion) {

        case 'A':
            break;
        case 'B':
            break;
        case 'C':
            break;
         case 'C':
             break;
         case 'D':
             break;
         case 'E':
             break;
         case 'F':
             break;
         case 'G':
             break;
         case 'H':
             break;
         case 'I':
             break;
         case 'J':
             break;
         case 'S':
             cout << "Saliendo del programa...\n";
             break;
         default:
             cout << "Opcion incorrecta!\n\n";
             break;
        }
    } while(opcion != 'S');

    return 0;
}
```

NORMAS PARA LA REALIZACIÓN DE LA PRÁCTICA

1. Las prácticas se realizarán individualmente o en grupos de dos alumnos que deberán ser los mismos para las dos prácticas de la asignatura.
2. Debe entregarse un fichero comprimido incluyendo todos los ficheros fuente del proyecto C++ y el documento descrito en el punto 6. Se subirá un fichero por grupo a la plataforma **antes de las 23:59 del día 12 de noviembre**. El nombre del fichero será el nombreapellido1apellido2 de uno de los miembros del grupo.
3. La asistencia y realización de la defensa, es individual y obligatoria para todos los alumnos del grupo, siendo necesario traer la memoria en papel para la realización de la misma. En el caso de no asistir, la práctica se calificará con un suspenso (0).
4. En la **defensa** de la práctica **se verificará la autoría de la práctica entregada y será calificada con APTO/NO APTO, siendo necesaria la calificación de APTO para poder ser evaluado de la práctica**. La defensa se realizará a la hora y en el lugar indicado por el profesor.
5. La entrega de prácticas copiadas supondrá el suspenso de la asignatura en esta convocatoria para todos los alumnos implicados.
6. La documentación que se entregará impresa, junto con la defensa, deberá tener al menos los siguientes apartados:
    a. Nombre y DNI de los alumnos del grupo.
    b. Detalles y justificación de la implementación:
        b.1 Especificación concreta de la interfaz de los TAD´s implementados:
            b.1.1 TAD`s creados.
            b.1.2 Definición de las operaciones del TAD (Nombre, argumentos y retorno).
        b.2 Solución adoptada: descripción de las dificultades encontradas.
        b.3 Diseño de la relación entre las clases de los TAD implementados.
            b.3.1 Diagrama UML.
            b.3.2 Explicación de los métodos más destacados.
            b.3.3 Explicación del comportamiento del programa.
        b.4 Código fuente.
        b.5 Bibliografía.