```
#include <fstream>
#include <algorithm>
#include "loadMembresia.h"
#include <string.h>
#include "Vinos/Vinos.h"

using namespace std;

std::string getIDDelUsuarioDeLaMembresia(Membresia* m) {
    return m->id_usuario;
}

std::string getIDVinoDeLaMembresia(Membresia* m, int iVino) {
    std::string id_Vino;

    switch (iVino) {
    case 0:
        id_Vino = m->id_vino_1;
        break;
    case 1:
        id_Vino = m->id_vino_2;
        break;
    case 2:
        id_Vino = m->id_vino_3;
        break;
    case 3:
        id_Vino = m->id_vino_4;
        break;
    case 4:
        id_Vino = m->id_vino_5;
```

```
        break;
    case 5:
        id_Vino = m->id_vino_6;
        break;
    default:
        id_Vino = "Error";
        std::cout << "No existe el vino que busca" << std::endl;
        break;
    }

    return id_Vino;
}

string getYearOfList(void *lista) {
    ELEMENTO innerList;
    obtenerElementoInicialDeLaLista((Lista*)lista, innerList);

    return ((Membresia*)innerList)->anio;
}

//Crea una nueva lista e inserta una membresia al final
Lista* createNewYearList(Membresia *membresia, Lista* lista) {
    Lista *listaAnio = crearLista();
    insertarElementoAlFinalDeLaLista(listaAnio, membresia);

    return listaAnio;
}

/*
  pre : str debe contener los datos necesarios para cargar una membresia.
```

```
  post: Se limpia y separa str en cada uno de los datos para la membresia.


  str : Cadena a la cual se va a quitar espacios, tabs y luego separarla.
  del : Cadena que separa los datos de str.
  return Array[8] string con los datos de la membresia [id_usuario, mes, anio, id_vino_1,
id_vino2, ...]
*/
std::string* splitStrByChar(std::string str, std::string del) {
    int start;
    int endStr;
    int position = 0;
    //limpio los datos de espacios, tabs & ;
    str.erase(std::remove(str.begin(), str.end(), ' '), str.end());
    str.erase(std::remove(str.begin(), str.end(), '\t'), str.end());
    str.erase(std::remove(str.begin(), str.end(), ';'), str.end());

    start = 0;
    endStr = str.find(del);
    //array que contiene los datos separados
    std::string* values = new std::string[9];

    //Separo los datos por el caracter del param del
    while (endStr != -1) {
        values[position] = str.substr(start, endStr - start);
        start = endStr + del.size();
        endStr = str.find(del, start);
        position++;
    }
    values[8] = str.substr(start, endStr - start);
    return values;
}
```

```
//Carga los datos del archivo en la lista de membresias
void readFileAndLoad(std::string path, Lista *lista) {
   std::ifstream archivo(path.c_str());
   std::string linea;

   while (getline(archivo, linea)) {
      if(linea.length() > 1) {
         std::string *valores = splitStrByChar(linea, "-");

         //crear membresia
         Membresia *membresia = new Membresia();

         membresia->id_usuario = valores[0];
         membresia->mes = valores[1];
         membresia->anio = valores[2];
         membresia->id_vino_1 = valores[3];
         membresia->id_vino_2 = valores[4];
         membresia->id_vino_3 = valores[5];
         membresia->id_vino_4 = valores[6];
         membresia->id_vino_5 = valores[7];
         membresia->id_vino_6 = valores[8];

         //Si la lista esta vacia, se crea un nodo con el año y se inserta
         if(listaEstaVacia(lista)) {
            Lista *yearList = createNewYearList(membresia, lista);
            insertarElementoAlFinalDeLaLista(lista, yearList);
         } else {
            bool insertado = false;
```

```
            //Se recorre la lista mientras i sea menor al tamaño y no se haya insertado el
dato
            for(int i = 0; i < getCantidadDeElementosEnLaLista(lista) && !insertado; i++)
{
               ELEMENTO innerElemento;
               obtenerElementoDeLaLista(lista, i, innerElemento);


               //Se comparan los años de la membresia creada y la de la lista, si coinciden
se inserta y se setea insertado= true
               if(membresia->anio.compare(getYearOfList((Lista*)innerElemento)) == 0)
{
                  Lista *innerList = (Lista*)innerElemento;
                  insertarElementoAlFinalDeLaLista(innerList, membresia);
                  insertado = true;
               }
            }


            //Si al terminar de recorrer la lista no se inserto el dato, significa que no existe
ese año, entonces se lo crea y agrega.
            if(!insertado) {
               Lista *yearList = createNewYearList(membresia, lista);
               insertarElementoAlFinalDeLaLista(lista, yearList);
            }
         }
      }
   }
}

void showMembresiaList(Lista *listaMembresia) {
   for(int i = 0; i < getCantidadDeElementosEnLaLista(listaMembresia); i++) {
      ELEMENTO innerElemento;
      obtenerElementoDeLaLista(listaMembresia, i, innerElemento);
```

```
        Lista *innerList = (Lista*)innerElemento;


        cout << "Año: " << getYearOfList(innerList) << " Cantidad: " <<
getCantidadDeElementosEnLaLista(innerList) << endl;
        for(int x = 0; x < getCantidadDeElementosEnLaLista(innerList); x++) {
            ELEMENTO membresia;
            obtenerElementoDeLaLista(innerList, x, membresia);


            cout << ((Membresia*)membresia)->id_usuario << "\t";
            cout << ((Membresia*)membresia)->mes << "/" << ((Membresia*)membresia)-
>anio << "\t";
            cout << ((Membresia*)membresia)->id_vino_1 << "\t";
            cout << ((Membresia*)membresia)->id_vino_2 << "\t";
            cout << ((Membresia*)membresia)->id_vino_3 << "\t";
            cout << ((Membresia*)membresia)->id_vino_4 << "\t";
            cout << ((Membresia*)membresia)->id_vino_5 << "\t";
            cout << ((Membresia*)membresia)->id_vino_6 << endl;
        }
        cout << '\n';
    }
}
```