```cpp
#include <iostream>
#include "RankingVinos.h"
#include "Lista.h"
#include "infoMensual.h"
#include "vino.h"
#include "cliente.h"
#include "RankingVarietales.h"
#include "string.h"
#include <iomanip>

using namespace std;

void rankingDeVarietales(Lista seleccionados, Lista clientes, Lista vinos){
    Lista rankingRango1 = rankingPorRangoEtario(seleccionados, clientes, vinos, 0, 30);
    Lista ordenada=crearLista();
    ordenarListaAuxVarietales(rankingRango1,ordenada);
    cout<<"Ranking de varietales de menores de 30:"<<endl;
    mostrarRanking(ordenada);
    Lista rankingRango2 = rankingPorRangoEtario(seleccionados, clientes, vinos, 30, 50);
    Lista ordenada1=crearLista();
    ordenarListaAuxVarietales(rankingRango2,ordenada1);
    cout<<"Ranking de varietales de entre 30 y 50:"<<endl;
    mostrarRanking(ordenada1);


    Lista rankingRango3 = rankingPorRangoEtario(seleccionados, clientes, vinos, 50, 1000);
    Lista ordenada2=crearLista();
    ordenarListaAuxVarietales(rankingRango3,ordenada2);
    cout<<"Ranking de varietales de mayores de 50:"<<endl;
    mostrarRanking(ordenada2);

}
//rankingPorRangoEtario ?
Lista rankingPorRangoEtario(Lista seleccionados, Lista clientes,Lista vinos, int valorEtarioInf, int valorEtarioSup) {
    Lista listaResultante = crearLista();
    Nodo nodoActual = seleccionados->inicio;

    while (nodoActual != NULL){
        infoMesual seleccionActual = (infoMesual) nodoActual->dato;
        int edadClienteActual = traerEdadCliente(clientes, getsIdUsuarioInfo(seleccionActual));

        if (edadClienteActual > valorEtarioInf && edadClienteActual < valorEtarioSup){

            Vino vino1 = traerVino(vinos, getsVino1(seleccionActual));
            Vino vino2 =traerVino(vinos, getsVino2(seleccionActual));
            Vino vino3 =traerVino(vinos, getsVino3(seleccionActual));
            Vino vino4 =traerVino(vinos, getsVino4(seleccionActual));
            Vino vino5 =traerVino(vinos, getsVino5(seleccionActual));
            Vino vino6 =traerVino(vinos, getsVino6(seleccionActual));

            agregarVarietal(listaResultante,getsVarietal(vino1));
            agregarVarietal(listaResultante, getsVarietal(vino2));
            agregarVarietal(listaResultante,getsVarietal(vino3));
            agregarVarietal(listaResultante, getsVarietal(vino4));
            agregarVarietal(listaResultante, getsVarietal(vino5));
            agregarVarietal(listaResultante, getsVarietal(vino6));

        }
        nodoActual = nodoActual->siguiente;
    }
```

```
61
62      return listaResultante;
63  }
64
65  void agregarVarietal(Lista lista,char varietalSeleccionado[]){
66      Nodo nodoActual = lista->inicio;
67      bool encontrado = false;
68
69      while(nodoActual != NULL && !encontrado){
70          RankingVarietales ranking = (RankingVarietales) nodoActual->dato;
71
72          if(strcmp(ranking -> varietal, varietalSeleccionado )== 0 ){
73              encontrado = true;
74              ranking->cantidadSelecciones +=1;
75          }else{
76              nodoActual = nodoActual->siguiente;
77          }
78      }
79      if(!encontrado){
80          RankingVarietales nuevoVarietal = new RankingVarietalesStruct();
81          nuevoVarietal->cantidadSelecciones = 1;
82          strcpy(nuevoVarietal->varietal,varietalSeleccionado);
83          agregarNodo(lista, nuevoVarietal);
84      }
85  }
86
87  int traerEdadCliente(Lista clientes,int idBuscado){
88      Nodo nodoActual = clientes->inicio;
89      Cliente encontrado = NULL;
90      while(nodoActual != NULL && encontrado == NULL){
91          Cliente clienteActual = (Cliente) nodoActual->dato;
92          if(getsIdUsuario(clienteActual) == idBuscado){
93              encontrado = clienteActual;
94          }
95          nodoActual = nodoActual->siguiente;
96      }
97      return getsEdad(encontrado);
98  }
99
100 Vino traerVino(Lista vinos,int idBuscado){
101     Nodo nodoActual = vinos->inicio;
102     Vino encontrado = NULL;
103     while(nodoActual != NULL && encontrado == NULL){
104         Vino vinoActual = (Vino) nodoActual->dato;
105         if(getsVino(vinoActual)== idBuscado){
106             encontrado = vinoActual;
107         }
108         nodoActual = nodoActual->siguiente;
109     }
110     return encontrado;
111 }
112
113 //void ordenarRanking(Lista lista){
114 //
115 //}
116
117 void mostrarRanking(Lista ranking){
118
119     Nodo nodoActual = ranking->inicio;
120      cout << setw( 70 ) << setfill( '-' ) << '\n' << setfill( ' ' );
121     cout << "| "<< left << setw( 14 ) << "Posicion"<< "|" << right << setw( 1) << "
Varietal"  << right << setw( 21 )<< "|"<< "Cantidad Selecionado" << " |"<<endl;
122     cout << setw( 70 ) << setfill( '-' ) << '\n' << setfill( ' ' ) << '\n';
123     while(nodoActual != NULL){
124             RankingVarietales puestoActual = (RankingVarietales) nodoActual->dato;
125             int resultado=strlen(puestoActual->varietal);
```

```
126             cout<<"Nro en lista: "<<nodoActual->nro<<"  varietal: "<<puestoActual->
varietal<<right << setw(38-resultado)<<" Cant.selecciones: "<<puestoActual->
cantidadSelecciones<<std::endl;
127
128             nodoActual = nodoActual->siguiente;
129     }
130     cout<<"\n"<<endl;
131
132 }
133
134 void ordenarListaAuxVarietales(Lista lista, Lista ordenada){
135     Nodo actual;
136     Nodo siguiente;
137     actual=lista->inicio;
138     Nodo auxNodo;
139     int nro;
140
141     while (actual!=NULL )
142     {
143         nro=actual->nro;
144         siguiente=actual->siguiente;
145         RankingVarietales auxMaximo=(RankingVarietales)actual->dato;
146         while( siguiente !=NULL )
147         {
148             RankingVarietales rankig13=(RankingVarietales)siguiente->dato;
149           if(auxMaximo->cantidadSelecciones<rankig13->cantidadSelecciones)
150             {
151             auxMaximo=rankig13;
152             nro=siguiente->nro;
153             }
154             siguiente=siguiente->siguiente;
155
156         }
157         RankingVarietales nuevo = new RankingVarietalesStruct;
158         auxNodo=nodoSeleccionado(lista,nro);
159         RankingVarietales varietalesAnt=(RankingVarietales)auxNodo->dato;
160         nuevo->cantidadSelecciones=varietalesAnt->cantidadSelecciones;
161         strcpy(nuevo->varietal,varietalesAnt->varietal);
162
163         agregarNodo(ordenada,nuevo);
164         eliminarNodo(lista,nro);
165
166         actual=lista->inicio;
167
168
169     }
170 }
```