EJERCICIOS INTEGRADORES

```
/*(26) Un buque de carga traslada 100 contenedores a tres diferentes puertos del país.
Los puertos se identifican con los números 1, 2 y 3.
De cada contenedor que el buque traslade se registran los siguientes datos:
- Identificación del contenedor
- Peso del contenedor en kg
- Puerto de arribo(un valor de 1 a 3).
Se pide calcular e informar:
1) El peso total que el buque debe trasladar
2) La identificación del contenedor de mayor peso
3) La cantidad de contenedores que debe trasladar a cada puerto*/

struct Contenedor
{int id;
 int peso;
 int puerto; //1, 2, 3}

int pesototal (Contenedor c[])
{ int pesot=0;
  for (i=0; i<100; i++)
  {pesot= c[i].peso + pesot}
return pesot;}

int contenedormaspesado (Contenedor c[])
{ int mayorpeso=0;
  int idmayorpeso;
  for (i=0; i<100; i++)
  {if(c[i].peso>mayorpeso)
     {mayorpeso=c[i].peso;
      c[i].id=idmayorpeso;}}
 return idmayorpeso;}

void cantapuerto (Contenedor c[])
{ int p1, p2, p3;
  p1=0;
  p2 =0;
  p3=0;
for (i=0; i<100; i++)
  { if (c[i].puerto == 1)
     {p1++;}
     else
      {if (c[i].puerto == 2)
        {p2++;}
        else
            {p3++;}}}
cout << "A puerto 1:" << p1 << endl;
cout << "A puerto 2:" << p2 << endl;
cout << "A puerto 3:" << p3 << endl;  }

int main()
{ Contenedor contenedores[100];
  cout << "Peso total:" << pesototal(contenedores[100]) << endl;
  cout << "Contenedor mayor peso:" << << endl;
  cout << "Cantidad de contenedores a cada puerto" << endl;
```

```
  cantapuerto(contenedores[]);
}
```

```
/*(32)Una compañía aérea desea emitir un listado con los movimientos mensuales de
sus M vuelos al exterior. Para ello cuenta con la siguiente información. De cada vuelo
realizado el número de vuelo, destino, y cantidad de asientos.
De cada pasajero el número de pasaporte y el importe que abonó por el pasaje en dólares.
Se pide emitir el siguiente

listado:
Nro. de Vuelo 9999 Destino: xxxxxxxxxxxxxxxx
 Nro. de Pasaporte Importe en u$s
 99999999 999.99
 99999999 999.99
 Total recaudado del vuelo: 99999.99
 % de Asientos Libres del vuelo 999.99
 % de Asientos Ocupados del vuelo 999.99
Total recaudado en el mes 999999.99
Cantidad de veces seguidas que se dieron vuelos completos 99
El número de vuelo que más recaudó 9999
*/

struct vuelo
{int numvuelo;
 char destino[25];
 int cantasientos;}

struct pasajeros
{ int pasaporte;
  int importeabonado;}

int cantidadpasajeros(pasajeros p[])
{int i;
 int cant;
for (i=0; i<170; i++)
{if (p[i].pasaporte != 0 || p[i].importeabonado != 0)
  {cant++;}}
return cant;}

float totallibre (pasajeros p[])
{ float totallibre;
(totallibre= cantidadpasajeros (p[170])*100)/170;
return totallibre;}

float totalocupado(pasajeros p[])
{float totalocupado;
(totalocupado= (170-cantidadpasajeros (p[170]))*100)/170;
return totallibre;}

float totalmes(vuelo v[], pasajeros p[])
{
return total;}

float recuadoporvuelo (vuelo v[], pasajeros p[])
{ float tot[150];
  float totalvuelo=0;
  for (i=0; i<150; i++)
```

```
   {for (i=0; i<170; i++)
   {totalvuelo = totalvuelo + p[i].importeabonado;}
   tot[i] = totalvuelo;}}

//No se hace la carga de datos. Trabajo con tamaño que conocemos en los arrays.
int main ()
{ vuelos v[150];
  pasajeros [170];
 cout << "Numero de vuelo:" << i << "Destino" << v[i].destino << endl;
 /* Nro. de Pasaporte Importe en u$s
 99999999 999.99
 99999999 999.99 */
 cout << "Total recaudado del vuelo: " << total(v[150], p[170]) << endl;
 cout << "Porcentaje libre: " << totallibre(p[170]) << endl;
 cout << "Porcentaje ocupado: " << totalocupado(p[170]) << endl;
 cout << "Total recaudado del mes: " << totaldelmes() << endl;
 cout << "Cantidad de veces seguidas de vuelos completos: " << total() << endl;
 cout << "Vuelo que más recaudó: " << total() << endl;
}
```

3. Canasta Familiar
Temas evaluados: Abstracción, archivos, arreglos y estructuras de datos enlazadas.
Definición del contexto
Ante la variedad de los precios de un mismo producto de la canasta familiar, según el valor impuesto por cada uno de los minoristas en los diferentes puntos de venta al público, se requiere informar a la comunidad el valor de los mismos y los tres comercios que ofrecen el mejor precio para cada uno de esos productos.
Dinámica de la selección de los comercios que ofrecen los mejores precios.
Cuando un producto integra la canasta familiar (compuesta por una cantidad N conocida a priori de productos), debe seleccionarse al menos los tres de menor precio. Detectado un producto con estas características deberá consignarse su precio y los datos del local que lo comercializa.
Problema
Se dispone de un archivo Productos.bin, de registros con todos los productos del rubro alimentación, de todos los locales analizados. Cada registro contiene ID_Producto (int), DirecciónVenta (cadena de 30 caracteres), precio (int), prioritario (1,0).
Se dispone de las siguientes funciones que puede utilizar sin desarrollar.
a) Nodo* insertarOrdenado(Nodo *& l, T x).
b) Nodo* buscarInsertar(Nodo *& l, T x), int buscarInsertarEnVector(T V[ ], T x).
c) T pop(Nodo* & l).
d) void push(Nodo* &p, T x).
Se pide
1. Defina y declare todas estructuras de datos necesarias para la resolución del problema justificando la selección
2. Codifique o diagrame la función cargarPrioritarios. Que recibe el archivo de los productos y retorna los prioritarios (valor 1 en el campo de ese nombre) en una estructura que los contenga ordenados por ID_producto y precio de venta creciente, de los al menos tres comercios que vendan los productos con precio mínimo conteniendo el precio y los datos del comercio de venta.
3. Codifique o diagrame la función listarProductos que liste los productos obtenidos en la función anterior y en el orden que se propuso