Parcial Algoritmos y Programación II - 7541 Curso Calvo (3° instancia)
29 de Junio de 2023
Padrón: Apellido y Nombre:

1) Indicar la salida por pantalla y escribir las sentencias necesarias para liberar correctamente la memoria.

```
int main(){
 int *A, *C, *F;
 int **B;
 char *D, *E;
 int H = 65 + ULTIMO_DIGITO_PADRON ;
 char G = 'E';
 F = new int[3];
 for (int i = 0; i < 3; i++) {
    F[i] = H + i;
 }
 H++;
 A = F;
 C = new int;
 (*C) = A[1];
 A = F + 2;
 cout << (*F) << (*C) << A[0] << endl;
 B = &C;
 A = C;
 C = F + 1;
 (**B) = H + 2;
 cout <<  (*C) <<  (**B) << (*A) << endl;
 D = (char*) A;
 E = (char*) *B;
 F[1] = 70;
 cout << (*D) << (*E) << (*A) << endl;
 E = &G;
 H++;
 G = (char) H;
 H++;
 (*A) = H;
 cout << (*E) << (*D) << G << endl;

 // liberar la memoria
 // ...
 return 0;
}
```

2. Implementar el método distribuirProductosEnSucursales de la clase Distribuidor a partir de las siguientes especificaciones:

```
class Distribuidor {

 public:
  /* post: remueve los Productos de la cola 'productosDisponibles' y los agrega a alguna Sucursal de
   *      'sucursalesExistentes' que tenga la categoria del Producto.
   *      Aquellos Productos que no pueden ser agregados a ninguna Sucursal se devuelven a la cola
   *      'productosDisponibles'.
   */
  void distribuirProductosEnSucursales(Cola<Productos*>* productosDisponibles, Lista<Sucursal*>* sucursalesExistentes);
}
```

```
class Sucursal {

 public:

  Sucursal();

  /* post: devuelve las categorias de productos que vende.
   */
  Lista<string>* obtenerCategorias();

  /* post: devuelve todos los Productos que tiene la sucursal.
   */
  Lista<Producto*>* obtenerProductos();

  /* post: libera todos los Productos.
   */
  ~Sucursal();
};
```

```
class Producto {

 public:

  /* post: Producto con el nombre y la categoría
   *      indicados.
   */
  Producto(string nombre, string categoria)

  /* post: devuelve el nombre del Producto.
   */
  string obtenerNombre();

  /* post: devuelve la categoría del Producto.
   */
  string obtenerCategoria();
}
```

4. Diseñar la especificación e implementar el **TDA IndicadorDeCanal**. Debe proveer operaciones para:

- crear el IndicadorDeCanal recibiendo como parámetro el canal mínimo y el canal máximo.
- contar: devuelve la cantidad de canales activos. Un canal puede estar activo o no.
- obtener: devuelve el número de canal actual.
- seleccionar: cambia el canal actual por uno recibido como parámetro.
- avanzar: cambia el canal actual por el siguiente (+).
- retroceder: cambia el canal actual por el anterior (-).
- volver: cambia el canal actual por el previo.

Ejemplificar el uso de la clase IndicadorDeCanal.

Para aprobar es necesario tener al menos el 60% de cada uno de los ejercicios correctos y completos.
Para cada método escribir pre y post condición, si recibe argumentos y cuáles, y si retorna un dato y cuál. De faltar ésto, se considerará el código incompleto.
Duración del examen : 3 horas