

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;
    char G;
    int H;

    H = 70;
    G = (char) H;
    A = new int;
    (*A) = H;
    F = A;
    C = F;
    (*A) = 65;
    cout << (*A) << (*C) << (*F) << endl;

    B = &C;
    C = new int;
    (**B) = H - 2;
    cout << (*A) << (*C) << (**B) << endl;

    E = (char*) C;
    D = (char*) (*B);
    cout << (*E) << (*D) << endl;

    (*E) = G;
    cout << (*E) << (*C) << endl;

    F = (int*) D;
    while ( (*C) > 0) {
        (*C)--;
        cout << (*C) << (*D) << (**B) << endl;
        (*F) = (*F) - (**B);
    }

    // liberar la memoria
    // ...

    return 0;
}
```

2. Implementar para la clase Lista<T> con una estructura simplemente enlazada el siguiente método, indicando pre y post:

```
T remove(unsigned int posicionElemento);
```

3. Implementar el método buscarMensajeMasVotadoDelUsuario de la clase Moderador a partir de las siguientes especificaciones:

<pre>class Moderador { public:     /* post: busca en la lista 'foros' el Mensaje más votado del autor 'usuarioBuscado' dentro de un Foro que incluya la     * temática 'tematicaBuscada'.     */     Mensaje* buscarMensajeMasVotadoDelUsuarioSegunTematica(Lista&lt;Foro*&gt;* foros,   string usuarioBuscado, string tematicaBuscada); };</pre>	<pre>class Foro { public:     /* post: inicializa el Foro sin Mensajes asignados.     */     Foro(string nombre);      /* post: elimina todos los Mensajes del Foro     */     ~Foro();      /* post: devuelve el nombre del Foro.     */     string obtenerNombre();      /* post: devuelve todos los Mensajes del Foro.     */     Lista&lt;Mensaje*&gt;* obtenerMensajes();      /* post: devuelve las temáticas del Foro     */     Lista&lt;string*&gt;* obtenerTematicas(); };</pre>
	<pre>class Mensaje { public:     /* post: 'usuario' es el autor del mensaje con 'texto'     * como contenido, sin votos asociados.     */     Mensaje(string usuario, string texto);      /* post: devuelve el nombre del usuario autor del Mensaje.     */     string obtenerAutor();      /* post: devuelve el contenido del Mensaje.     */     string obtenerContenido();      /* post: suma un voto al Mensaje.     */     void votar();      /* post: devuelve la cantidad del Ingrediente.     */     unsigned contarVotos(); };</pre>

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

- Construir el Restaurante a partir de la cantidad de mesas que tiene. Cada mesa se identifica con un número.
- adicionarEnMesa: agrega el monto dado [\$] a la mesa indicada, dejándola abierta.
- cerrarMesa: devuelve el total de la mesa indicada y la deja libre.
- contarMesasLibres: cuenta la cantidad de mesas que están libres.
- totalizarRecaudacion: devuelve el total recaudado por el Restaurante.

**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**