Parcial Algoritmos y Programación II - 7541 Curso Calvo (1º instancia)
04 de Noviembre de 2013

Padrón: **Apellido y Nombre:**

Punteros: APROBADO - DESAPROBADO

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 = 66;
 char G = 'C';

 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 para la clase ListaSimplementeEnlazada el método

```
 /* pre : la lista tiene al menos tantos elementos como posicionElemento.
  * post: remueve el elemento localizado en la posición posicionElemento de la lista y lo devuelve.
  */
 T remover(unsigned int posicionElemento);
```

3. Implementar el método buscarLibrosNoPrestadosPorAutor de la clase Bibliotecario a partir de las siguientes especificaciones:

```
class Bibliotecario {
 public:
   /* post: busca en estantes aquellos Libros que no están prestados y que uno de sus autores es autorBuscado.
    *       Devuelve una nueva Lista con todos los Libros en esta condición. */
   Lista<Libro*>* buscarLibrosNoPrestadosPorAutor(Lista<Estante*>* estantes, string autorBuscado);
};
```

```
class Estante {

 public:

   /* post: inicializa el Estante sin Libros asignados.
    */
   Estante();

   /* post: elimina todos los Libros que tiene el Estante.
    */
   ~Estante();

   /* post: devuelve todos los Libros asignados al Estante.
    */
   Lista<Libro*>* getLibros();

   /* post: devuelve aquellos Libros que fueron prestados,
    *      pero que están asignados al Estante.
    */
   Lista<Libro*>* getLibrosPrestados();
};
```

```
class Libro {

 public:

   /* post: inicializa el Libro sin autores
    *      asociados y con el titulo indicado.
    */
   Libro(string titulo);

   ~Libro();

   /* post: devuelve el titulo del Libro.
    */
   string getTitulo();

   /* post: devuelve los nombres de los autores del Libro.
    */
   Lista<string>* getAutores();
};
```

4. Diseñar la especificación e implementar el TDA **Ascensor**. Un Ascensor se debe crear recibiendo como parámetro la cantidad de pisos por los que se mueve (sin considerar la planta baja). Debe proveer operaciones para:

- devolver el número de piso en el que se encuentra, considerando 0 como la planta baja.
- llamar desde un piso: debe moverlo y devolver la cantidad de pisos que el ascensor se movió para llegar al piso indicado.
- devolver la cantidad total de pisos que el ascensor subió y bajó.
- devolver la cantidad de veces que fue a un piso.
- Devolver el piso al que más veces se llamó.
- El ascensor puede estar en Funcionamiento o no (roto)

Hacer el main

Los alumnos que tienen aprobado el parcialito de punteros no deben realizar el ejercicio 1.
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 : 2,5 horas