Parcial Algoritmos y Programación II - 7541 Curso Calvo (2º instancia) 18 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;
    char G;
    int H;

    G = 'B';
    D = &G;
    E = D;
    (*E) = 'C';
    cout << (*D) << G << (*E) << endl;

    H = 64;
    A = new int[2];
    (*A) = H;
    B = &A;
    C = A + 1;
    F = A;
    F[1] = 65;
    cout << (**B) << (*F) << (*C) << endl;

    E = (char*) (F + 1);
    G = 'D';
    (*C) = 66;
    cout << (*D) << (*E) << F[1] << endl;

    D = (char*) (*B);
    A[1] = 67;
    G = 'X';
    cout << (*E) << (*D) << (*C) << endl;

    // liberar la memoria
    // ...

    return 0;
}
```

2. Implementar para la clase ListaDoblementeEnlazada 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.
 */
template<class T>
T remover(unsigned int posicionElemento);
```

3. Implementar el método buscarReceteasPosibles de la clase Chef a partir de las siguientes especificaciones:

```
class Chef {
  public:
    /* post: busca en la lista recetas aquellas Recetas que pueden ser preparadas con los Ingredientes que se encuentran
     *       en ingredientesDisponibles. Considera que todos los Ingredientes de la Receta deben estar en
     *       ingredientesDisponibles comparando su nombre y validando que la cantidad sea la suficiente.
     *       Devuelve una nueva Lista con todos las Recetas en esta condición. */
    Lista<Receta*>* buscarRecetasPosibles(Lista<Receta*>* recetas, Lista<Ingrediente*>* ingredientesDisponibles);
};
```

```
class Receta {

  public:

    /* post: inicializa la Receta sin Ingredientes asignados.
     */
    Receta(string nombre);

    /* post: elimina todos los Ingrediente de la Receta.
     */
    ~Receta();

    /* post: devuelve todos los Ingredientes de la Receta.
     */
    Lista<Ingrediente*>* getIngredientes();

    /* post: devuelve el nombre que identifica a la Receta.
     */
    string getNombre();
};
```

```
class Ingrediente {

  public:

    /* post: queda inicializado con el nombre y cantidad
     *       indicados.
     */
    Ingrediente(string nombre, float cantidad);

    ~Ingrediente();

    /* post: devuelve el nombre del Ingrediente.
     */
    string getNombre();

    /* post: devuelve la cantidad del Ingrediente.
     */
    float getCantidad();
};
```

4. Diseñar la especificación e implementar el TDA **Entrenamiento**. Un Entrenamiento se debe crear recibiendo como parámetro la cantidad de días de duración. Debe proveer operaciones para:

- devolver la duración (en días)
- registrar la cantidad de kilómetros corridos (por ejemplo 1.5) en uno de los días (identificado por número de orden).
- anular el entrenamiento en uno de los días.
- devolver la cantidad de kilómetros corridos en un día.
- devolver la cantidad de kilómetros corridos en total.
- devolver el promedio de kilómetros corridos por día (considerando solo aquellos días en los no fue anulado).
- devolver la cantidad de días que corrió más que el promedio.

Nota: No debe ser posible registrar más de una vez la cantidad de kilómetros corridos en un día particular.

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 : 3 horas