Parcial Algoritmos y Programación II - 7541 Curso Calvo (1º instancia) 31 de mayo de 2019

Padrón: **Apellido y Nombre:**

Punteros: APROBADO – DESAPROBADO **TDA:** APROBADO – DESAPROBADO

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

```
int main(){
  int *A, *C, *F;
  char *D, *E;
  char **B;
  char G;
  int H;

  H = 66 + ULTIMO_DIGITO_PADRON;
  G = 'A';
  A = new int;
  (*A) = H;
  C = &H;
  F = A;
  H++;
  cout << H << (*A) << (*C) << (*F) << endl;

  D = new char[4];
  E = D + 1;
  B = &D;
  D[0] = G;
  D[1] = 'C';
  cout << (*D) << (*E) << (**B) << endl;

  E[1] = *(D + 1);
  E[2] = G;
  E = &G;
  G = 'B';
  cout << (*B)[2] << (*E) << *(D + 2) << endl;

  E = D;
  D = (char*) F;
  (*A)--;
  cout << (**B) << (*A) << (*D) << endl;

  // liberar la memoria

  return 0;
}
```

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

```
void insertar(unsigned int posicionElemento, T elemento);
```

3. Implementar el método `buscarAtracciones` de la clase `AgenteDeViajes` a partir de las siguientes especificaciones:

```
class AgenteDeViajes {

  public:
    /* post: busca en 'lugaresDisponibles' aquellas Atracciones que estén en lugares no visitados y cuya duración total
     *       (la suma de las duraciones de todos las Atracciones) no supere 'minutosDisponibles'.
     /
    Lista<Atraccion*>* buscarAtracciones(Lista<Lugar*>* lugaresDisponibles,
                                         Lista<string>* nombresDeLugaresYaVisitados, unsigned int minutosDisponibles);
}
```

```
class Lugar {

  public:

    /* post: Lugar identificado por 'nombre' y sin Atracciones.
     */
    Lugar(string nombre);

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

    /* post: devuelve todas las Atracciones que tiene.
     */
    Lista<Atraccion*>* obtenerAtracciones();

    /* post: libera todas las Atracciones del Lugar.
     */
    ~Lugar();
};
```

```
class Atraccion {

  public:

    /* post: Atracción identificada por 'nombre',
     *       cuya duración es 'minutos'
     */
    Atraccion(string nombre, unsigned int minutos)

    /* post: devuelve el nombre de la atracción.
     */
    string obtenerNombre();

    /* post: devuelve la duración en minutos.
     */
    unsigned int obtenerDuracion();

}
```

4. Diseñar la especificación e implementar el **TDA Cronómetro** con las siguientes operaciones:

- Crearlo
- Iniciar la medición de un nuevo intervalo de tiempo.
- Terminar la medición del intervalo de tiempo.
- Consultar la cantidad de segundos que tiene el intervalo de tiempo medido.
- Devolver el máximo intervalo de tiempo medido [segundos].

Se dispone de la función: **unsigned int** upTime(); que devuelve la cantidad de milisengudos transcurridos desde que se encendió la máquina.

Los alumnos que tienen aprobado el parcialito de punteros y/o TDA no deben realizar el ejercicio 1 y/o 4 respectivamente.
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

```
***0***
67666766
ACA
CBC
A65A
***1***
68676867
ACA
CBC
B66B
***2***
69686968
ACA
CBC
C67C
***3***
70697069
ACA
CBC
D68D
***4***
71707170
ACA
CBC
E69E
***5***
72717271
ACA
CBC
F70F
***6***
73727372
ACA
CBC
G71G
***7***
74737473
ACA
CBC
H72H
***8***
75747574
ACA
CBC
I73I
***9***
76757675
ACA
CBC
J74J
***TODOS***
delete A
delete[] E
```

Los alumnos que tienen aprobado el parcialito de punteros y/o TDA no deben realizar el ejercicio 1 y/o 4 respectivamente.
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