Parcial Algoritmos y Programación II - 7541 Curso Calvo (1º instancia) 1 de noviembre 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 `buscarTallerConTurnoLibre` de la clase `Mecanico` a partir de las siguientes especificaciones:

```
class Mecanico {

  public:
    /* post: devuelve una nueva Lista con todos aquellos Talleres existentes en 'talleresDisponibles' que atiendan
     *       la especialidad indicada y que tengan al  menos un Turno no ocupado entre 'horaDesde' y 'horaHasta'.
     */
    Lista<Taller*>* filtrarTaller(Cola<Taller*>* talleresDisponibles,
                                  string especialidad, unsigned int horaDesde, unsigned int horaHasta);
}
```

```
class Taller {

  public:

    Taller();

    /* post: devuelve todas las especialidades que atiende.
     */
    Lista<string>* obtenerEspecialidades();

    /* post: devuelve todos los Turnos que tiene.
     */
    Lista<Turno*>* obtenerTurnos();

    /* post: libera todos los Turnos.
     */
    ~Taller();
};
```

```
class Turno {

  public:
    /* post: Turno no ocupado entre la horas desde y
     * hasta.
     */
    Turno(unsigned int desde, unsigned int hasta)
    /* post: devuelve la hora de inicio del turno.
     */
    unsigned int obtenerHoraDesde();
    /* post: devuelve la hora de fin del turno.
     */
    unsigned int obtenerHoraHasta();
    /* post: indica si el turno está ocupado.
     */
    bool estaOcupado();
    /* post: marca el turno como ocupado.
     */
    void ocupar();
}
```

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

- Crear el Pastillero recibiendo como parámetro la cantidad de pastillas disponibles y el tiempo mínimo [horas] entre una toma y la siguiente [1-24].
- Contar la cantidad de pastillas que restan tomar.
- Tomar una pastilla, recibiendo como parámetro la hora del día en la que se tomó [0-23].
- Calcular la hora [0-23] en la que se debe tomar la siguiente pastilla.
- Contar la cantidad de veces que fue tomada una pastilla habiendo pasado más tiempo que el indicado.

Ejemplificar el uso de la clase Pastillero.

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