Parcial Algoritmos y Programación II - 7541 Curso Calvo (2° instancia)
15 de junio de 2023
Padrón: **Apellido y Nombre:**
Punteros: APROB/DESAPROB **TDA:** APROB/DESAPROB **LISTAS:** APROB/DESAPROB

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

- Crear el Pastillero recibiendo como parámetro la cantidad de casilleros (un casillero por día) y la cantidad máxima de pastillas que puede almacenar cada uno.
- Contar la cantidad de pastillas que restan tomar a partir de un día.
- Contar la cantidad de pastillas que no se tomaron hasta un día.
- Agregar y quitar una pastilla al Pastillero indicando el día.
- Devolver el total de pastillas y el promedio por día.

Ejemplificar el uso de la clase Pastillero.

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*>* buscarTallerConTurnoLibre  (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();
}
```

Los alumnos que tienen aprobado algún punto, no deben volver a hacerlo.
Para aprobar es necesario tener al menos el 60 % de cada uno de los ejercicios correctos y completos.
En TDA no se puede utilizar listas y respetar todos los principios de TDA.
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 algún punto, no deben volver a hacerlo.
Para aprobar es necesario tener al menos el 60 % de cada uno de los ejercicios correctos y completos.
En TDA no se puede utilizar listas y respetar todos los principios de TDA.
Duración del examen: 3 horas