Parcial Algoritmos y Programación II - 7541 Curso Calvo (2° instancia)
9 de Junio de 2014

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 = 'B';

 A = new int[3];
 for (int i = 0; i < 3; i++) {
    A[i] = H + i;
 }

 H++;
 F = A;
 C = F + 1;
 B = &F;
 cout << (*A) << (**B) << (*C) << endl;

 H = (**B) - 2;
 F[1] = A[0] - 2;
 F = A + 2;
 (**B) = (**B) - 3;

 cout << (*A) << (**B) << (*C) << endl;

 E = (char*) (C + 1);
 D = (char*) (&H);
 A[2] = A[0] + 3;
 cout << (*D) << G << (*E) << endl;

 C = new int;
 (*C) = (**B);
 (*B) = C;
 D = (char*) F;
 (*C) = (*C) - 4;

 cout << (*D) << G << (*E) << endl;

 (*(A + 2)) = 65;
 D = (&G);
 G = 'C';
 (*E) = (*E) + 1;

 cout << (*D) << G << A[2] << endl;

 // Setencias para liberar la memoria

 return 0;
}
```

2. Desarrollar una implementación dinámica basada en templates del TDA Pila.

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

```
class Bibliotecario {

 public:
  /* post: busca en 'estantes' aquellos Libros que no tienen al menos todas las categorias que indica su Estante.
   *      Devuelve una nueva Lista con todos los Libros en esta condición. */
  Lista<Libro*>* buscarLibrosMalAsignados(Lista<Estante*>* estantes);
};
```

```
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*>* obtenerLibros();

  /* post: devuelve las categorias que al menos deben tener
   *      todos los Libros localizados en este Estante.
   */
  Lista<string>* obtenerCategorias();
};
```

```
class Libro {

 public:
  /* post: inicializa el Libro sin categorias
   *      asociadas y con el titulo indicado.
   */
  Libro(string titulo);

  ~Libro();

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

  /* post: devuelve los nombres de las categorias del libro.
   */
  Lista<string>* obtenerCategorias();
};
```

4. Diseñar la especificación e implementar el TDA **Examen**. Debe proveer operaciones para:

- crearlo indicando la cantidad total de ejercicios que tiene.
- puntuar un ejercicio, indicando su porcentaje de corrección [0.0, 100.0].
- contar la cantidad de ejercicios que tiene el examen.
- contar los ejercicios que ya fueron corregidos.
- calcular la nota, devolviendo la nota final del examen, comprendida en el rango [0.0, 10.0].
- calcular el puntaje promedio obtenido por ejercicio.

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