EIE
Escuela de
Ingeniería Eléctrica

Universidad de Costa Rica
Facultad de Ingeniería
Escuela de Ingeniería Eléctrica

UNIVERSIDAD DE COSTA RICA

IE-0217 Estructuras abstractas de datos y algoritmos para ingeniería

Laboratorio 5: Listas

M. Sc. Ricardo Román-Brenes - ricardo.roman@ucr.ac.cr

I-2019

Tabla de contenidos

1. Enunciado

Implementar las estructuras lineales de datos utilizando plantillas, herencia y POO en C++, siguiendo los lineamientos del archivo `List.h`.

```
1  // ...
2
3  template<Data, Position>
4  class List {
5      public:
6          List();
7          List(const List &orig)
8          virtual ~List();
9
10         virtual void emptyList() = 0;
11
12         virtual void insert(Data, Position) = 0;
13         virtual void insert(Data) = 0;
14         virtual void delete(Data) = 0;
15         virtual void delete(Position) = 0;
16
17         virtual Data getElement(Position) = 0;
18         virtual Position find(Data) = 0;
19         virtual Position next(Position) = 0;
20         virtual Position prev(Position) = 0;
21
22         virtual void print() = 0;
23
24         // ...
25
26     private:
```

```
27          int items;
28
29          // ...
30
31  };
32
33  // ...
```

Las estructuras que debe implementar recibirán en los argumentos de una plantilla dos elementos: el tipo de dato que almacenará (*element*) y el tipo que utilizará como posición (lo que determinará su implementación subyacente, (*int*, *SinglePosition* o *DoublePosition*)).

1. Lista con arreglos: en un archivo llamado `ArrayList.h` Esta clase debe heredar de la clase `List` y usar enteros como posición.
2. Lista con punteros: en un archivo llamado `SingleLinkedList.h` ~~y~~ `DoubleLinkedList.h`. Esta clase debe heredar de la clase `List`. Para la posición implemente una clase emplantillada llamada `SimplePosition o DoublePosition` que reciba el dato que almacenará.

Necesitará también implementar al menos dos clases más, `SinglePosition` y `DoublePosition` que representan posiciones de listas simplemente y doblemente enlazadas.

Una vez creadas las estructuras, implemente los siguientes algoritmos de búsqueda y ordenamiento, en un archivo aparte, y analice sus complejidades.

- Mergesort.
- Quicksort.
- ~~Selection sort.~~
- Búsqueda lineal.
- Búsqueda binaria.
- ~~Búsqueda Fibonacci.~~

Haga un programa de pruebas para su código.

2. Consideraciones

- Haga grupos de hasta 3 personas.
- Genere un reporte en LaTeX con sus conclusiones y adjunte el código fuente como apéndice.
- Suba su código y documentación (doxygen, README, INSTALL) al git respectivo de su grupo y el directorio del laboratorio.
- Recuerde que por cada día tardío de entrega se le rebajaran puntos de acuerdo con la formula: 4^d, donde $d > 1$ es la cantidad de días tardíos.