Trabajo Práctico
Lavadero Industrial
(pre-entrega)

Estructuras de Datos
Tecnicatura en Programación Informática
Universidad Nacional de Quilmes

1. Introducción

El objetivo de este trabajo es utilizar las estructuras de datos vistas en la materia para representar la información utilizada por un lavadero industrial. El lavadero en cuestión se encarga básicamente de lavar prendas y sábanas con una modalidad particular, las prendas son propiedad del lavadero. Es decir que el lavadero presta prendas y sábanas que luego entran en el circuito de lavado. Realizaremos una implementación en el lenguaje `C++` de lo mencionado.

2. Clientes del lavadero

Comenzaremos programando una estructura que nos permita contener a los clientes de forma de insertarlos, buscarlos y borrarlos. Para eso vamos a programar el TAD Map. Lo que vamos a pedir es que al insertar, buscar y borrar elementos. Las operaciones se realicen con orden de complejidad logarítmico en peor caso.

3. TADs a implementar

3.1. TAD Cliente

```
Cliente crearCliente(string cuit, string nombre);
string getCuit(Cliente c);
string getNombre(Cliente c);
void destroyCliente(Cliente& c);
```

3.2. TAD Map String Cliente

```
Map emptyM();
void addM(Map& m, Cliente c);
Cliente lookupM(Map& m, string cuit);
void removeM(Map& m, string cuit);
ArrayList domM(Map& m);
void destroyM(Map& m);
```

4. Representación del Map String Cliente

Dado que el Map debe tener tiempos logarítmicos en sus funciones, lo deberán implementar utilizando un BST de clientes, donde el criterio de ordenamiento va a estar dado por el cuit del

cliente.

5. Fecha de entrega de la pre-entrega

La fecha de pre-entrega es el 21/11. Esta pre-entrega consiste en implementar el TAD Cliente y el TAD Map String cliente.

6. Forma de entrega

1. La entrega deberá realizarse vía mail a la dirección tpi-ed-doc@listas.unq.edu.ar antes de las 00 hs del 21/11.
2. El asunto del correo debe tener la forma [pre-entrega-lavadero][Juan Perez][C1] siendo "Juan Perez" su nombre y "C1" su comisión.

7. Pautas de evaluación

7.1. Se pide

1. Definir implementaciones eficientes utilizando exactamente la representación propuesta.
2. Dar los órdenes de complejidad en peor caso y justificarlos.
3. Definir, en caso de existir, los invariantes de estas estructuras.

7.2. Se penaliza

1. Utilizar una representación distinta a la dada.
2. No indicar propósitos, precondiciones, órdenes de complejidad e invariantes de representación.
3. Entrega fuera de término.
4. Incumplimientos de los propósitos de cada función.
5. Violar barreras de abstracción.
6. Incoherencia en las subtareas definidas.
7. No respetar la forma de entrega establecida.
8. Que el código no pueda compilar.
9. Que el código introduzca memory leaks.
10. Que el código introduzca bucles infinitos.
11. Utilizar conceptos que exceden el alcance de la materia (recursos de C++ que no se han visto).