.UBAfiuba
FACULTAD DE INGENIERÍA

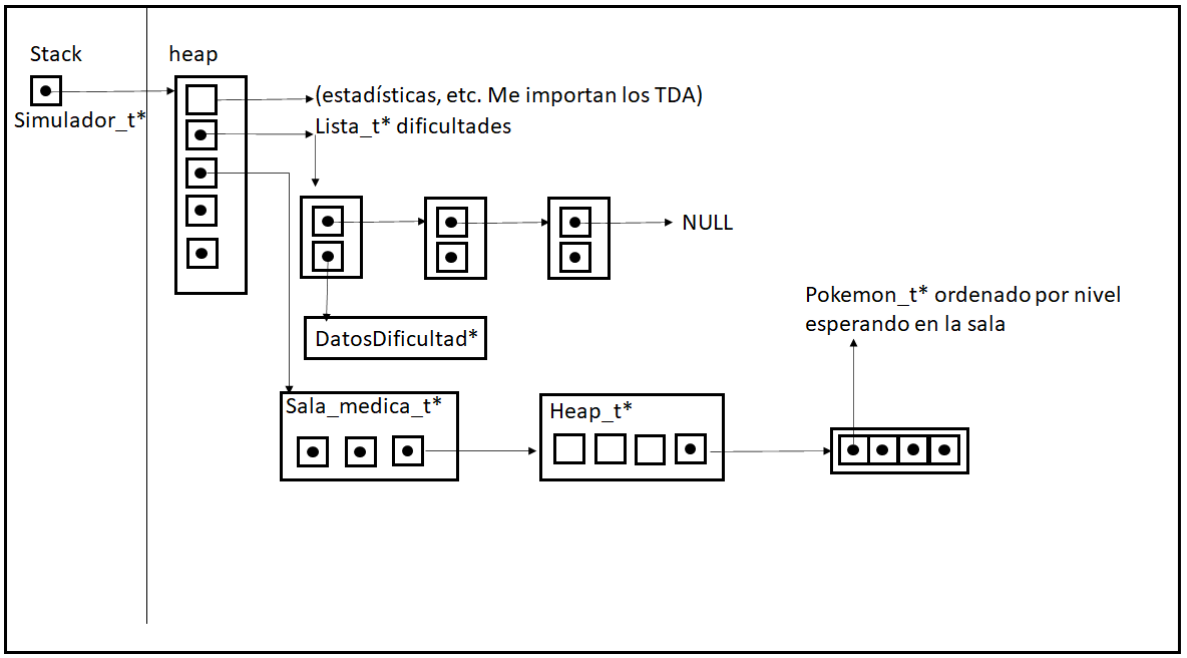
TP2- SIMULADOR POKEMÓN

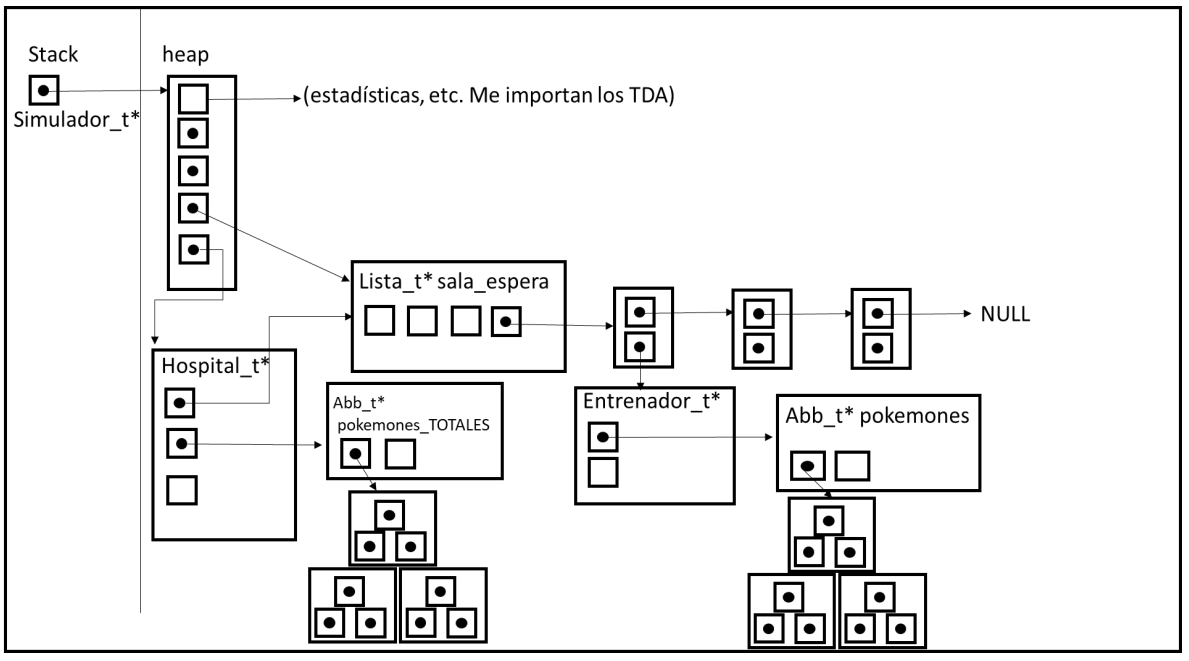
Alumno:	*Agustín de la Rosa*
Padrón:	*108275*
Email:	*adelarosa@fi.uba.ar*

Introducción y teoría:

Este trabajo se realizó con el objetivo de demostrar el correcto uso de los TDAS vistos en clase, para esto se usa el escenario de un simulador pokémon. El simulador está compuesto de un hospital, (el cual es el TP1),

Se crea el simulador habiendo pasado por parámetro el hospital, entonces nos queda la siguiente situación. *(Los cuadrados sin punto, no son punteros o si lo son, son datos sin interés para la explicación, como podría ser char* nombre).*





Parte 1: Se hace una lista simplemente enlazada para las dificultades y dentro de la sala médica, que significa el entrenador actualmente atendido, contiene punteros a el nombre del entrenador el pokemon actual y a un heap donde están esperando los demás pokemones, importante, ya ordenados por nivel.

Parte 2:: En la estructura del hospital, se tiene una lista simplemente enlazada hacía los entrenadores, donde cada entrenador, tiene un árbol binario de búsqueda sobre sus pokemones, ordenados de manera alfabética. entonces, los "menores" alfabéticamente van a la izquierda y los mayores a la derecha.

¿Por qué elegir estas estructuras?

Árbol binario de búsqueda: Lo importante de poner a los pokemones en un árbol, es que era requisito ordenar alfabéticamente, entonces, si yo elijo un árbol ya están ordenados de tal manera que si yo hago el recorrido INORDEN, puedo obtener los datos de menor a mayor sin haber usado algoritmos de ordenamiento ni un vector contiguo en memoria. Cada entrenador tiene a sus pokemones en un abb porque ya tenía todas las funciones escritas :).

Heap binario: Se elige heap binario antes que otro árbol binario por su poca complejidad algorítmica al extraer la raíz, como el heap es un algoritmo en un vector que se aplica siempre que se inserta un elemento, siempre está listo para extraer el primer elemento (raíz) que es el pokemon con menor nivel.

Lista simplemente enlazada: Se utiliza para no hacer vectores contiguos en memoria y así no suceda que realloc tenga que mover toda la cadena de memoria a otro lado.

Como ejecutar

Es necesario tener todos los archivos de la carpeta src, los cuales son:

.abb.c
.abb.h
.heap.c
.heap.h
.hospital.c
.hospital.h
.lista.c
.lista.h
.simulador.c
.simulador.h
.simulador_structs.h
.hospital_structs.h

Luego, me tome la molestia de armar una función en el main.c que recibe un hospital y lo simula (también lo destruye) simular_hospital(hospital_t* hospital); En el main también se incluyen las funciones de pruebas, el informe queda imprimido en pantalla hasta que empiece simular_hospital. Se puede usar el makefile para ejecutar perfectamente.