Facultad de Ciencias Exactas, Ingeniería y Agrimensura
Escuela de Ciencias Exactas y Naturales
Departamento de Computación
Estructuras de Datos y Algoritmos I

2do Parcial

1. Imagine que tiene implementada una cola genérica, `Queue`, en la que que puede almacenar valores de `void *`. Sus operaciones implementadas son:

- `Queue *queue_create()` que crea una cola.
- `int queue_is_empty(Queue *)` que determina si una cola está vacía.
- `void queue_enqueue(Queue *, void *)` que agrega un valor a la cola.
- `void *queue_front(Queue *)` que devuelve el elemento en primera posición.
- `Queue *queue_dequeue(Queue *)` que quita el elemento en primera posición.
- `void queue_destroy(Queue *)` que destruye una cola.

En la implementación dada en clase de árboles binarios, introduciamos una función

```
void btree_foreach(BTree *root, VisitorFuncInt visitor, void *extra_data)
```

donde `VisitorFuncInt` estaba definido como

```
typedef void (*VisitorFuncInt)(int, void *)
```

Esta función recorría los datos de un árbol, pero no permitia modificarlos, ya que el visitador recibia un entero que era una copia del elemento del árbol.
En este ejercicio, deberá:

a) Proponer una modificación de `VisitorFuncInt`, de nombre `VisitorFuncIntPtr`, que permita la modificación del entero que reciben las funciones.

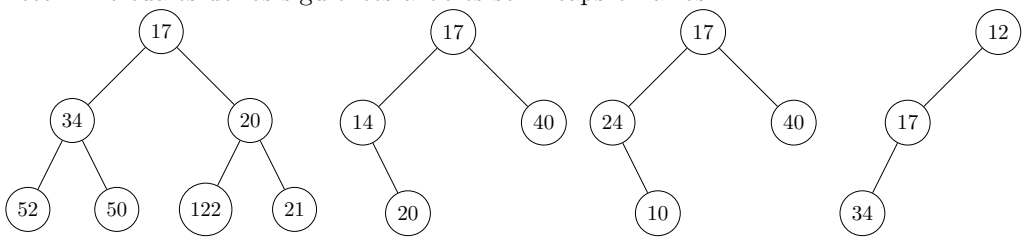
b) Escribir una implementación de `btree_foreach` que haga un recorrido "primero por extensión" (o "por niveles"), y vaya aplicando una fc. `VisitorFuncIntPtr`.

c) Escribir una función `void btree_replace(BTree *, int)` que dado un árbol y un entero, reemplace todos los datos de ese árbol por el entero dado. Utilice la función definida en el item anterior.

2. Explique qué tipo tiene una función hash, de qué sirve en una hash table, y enumere condiciones ideales que esta función debería satisfacer.

3.

a) Determine cuáles de los siguientes árboles son heaps binarios:



b) Imagine que crea un heap binario nuevo. Luego inserta los elementos: 10, 8, 11, 9, 12, y 2 (en este orden), según el algoritmo visto en clase para la función `bheap_insert`. Dibuje el heap resultante en forma de árbol y en forma de arreglo.

c) Al heap resultante del item anterior, le aplica dos veces la función `bheap_erase_minimum` según el algoritmo visto en clase. Dibuje nuevamente el heap resultante en forma de árbol y en forma de arreglo.