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

1er Parcial

1. Ya hemos visto cómo implementar colas y pilas desde cero, utilizando tanto arreglos como estructuras enlazadas. En este ejercicio, asumiremos que ud. ya ha implementado colas y desea implementar pilas, pero esta vez en lugar de empezar desde cero, utilizará las colas ya implementadas: en concreto, utilizará dos colas para implementar una pila (de manera posiblemente ineficiente).
Si `XQueue` es la estructura utilizada para implementar las colas, ud. deberá dar una implementación de `xstack_create`, `xstack_push`, `xstack_pop` y `xstack_is_empty` para la estructura:

```
typedef struct {
  XQueue *q1;
  XQueue *q2;
} XStack;
```

de manera que `XStack` y las respectivas funciones implementen una pila (las funciones `xstack_top` y `xstack_destroy` no es necesario hacerlas).

2. Imagine que debe llevar los DNIs y nombres de cada persona registrada en un sistema. Para esto, se propone utilizar una hashtable con números de DNI como claves y nombre como valores.

a) Suponiendo que un DNI lo implementa como un número entero sin signo, escriba una función hash para estas claves (asuma que la función hash recibe como argumento adicional la cantidad de cubetas). Enumere las condiciones ideales que una función hash debe tener, y argumente brevemente por qué la función propuesta es aceptable.

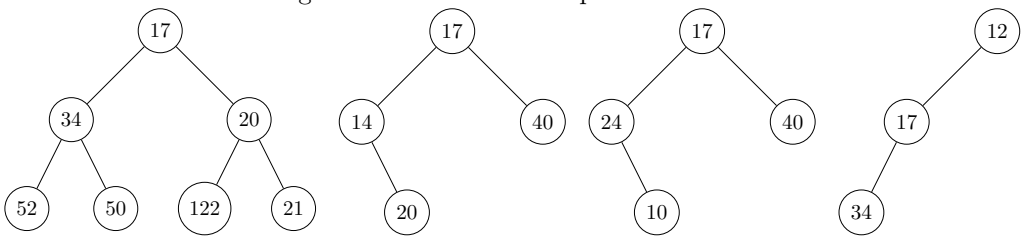
b) Escriba el código correspondiente para agregar a una hashtable `ht` la información:

$$18181612 \mapsto \text{“Espinosa Ricardo”}$$
$$18181615 \mapsto \text{“Perez Manuel”}$$
$$12181612 \mapsto \text{“Martínez Juan”}$$

c) Haga un esquema de cómo quedaria una tabla hash luego de agregarle los items solicitados en el ítem anterior, suponiendo que originalmente estaba vacía.

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.

4. Utilizando la función `btree_foreach`, escriba una función que imprima los elementos de un árbol y calcule el mínimo elemento del mismo en una sola pasada. Si no fuera posible, justifique y provea el código para hacerlo en dos pasadas.