Guía de Ejercicios

Guía de Ejercicios

Braulio A. Quiero Hernández

- Resuelva los ejercicios de programación en Lenguaje **C++**.

Filas

- Se recomienda realizar los ejercicios en la función main y también modularizarlos como una función independiente.

Utilizando solo operaciones de filas y pilas

1. Invierta una fila.
2. Tome 2 filas de distinto tamaño y entregue sus elementos intercalados.
3. Suponga que 3 filas guardan las edades de las personas. Construya un programa que atienda (muestre en pantalla) primero a la persona de mayor edad de las filas.

Listas

Dada la interfaz de listas doblemente enlazadas se solicita **construir las operaciones** a continuación e indicar el **orden de complejidad** en el peor de los casos.

En caso de no existir la firma, deduzca los parámetros de entra y salida necesarios.

1. `add`
    1. Que añade un elemento al principio (por _head) de la lista.
    2. Que añade un elemento al final (por _tail) de la lista .
    3. Que añade un elemento en la k-ésima posición de la lista. Suponga que el elemento apuntado por _head tiene la posición 0.
    4. Que añade un elemento
2. `remove`
    1. Que elimina un elemento al principio de la lista.
    2. Que elimina un elemento al final de la lista.
    3. Que elimina un elemento en la k-ésima posición de la lista.
    4. Que elimina la primera ocurrencia de un elemento.
    5. Que elimina todas las ocurrencias de un elemento.
3. `get`
    1. Que obtiene el primer elemento de la lista.
    2. Que obtiene el último elemento de la lista.
    3. Que obtiene el elemento en la k-ésima posición de la lista.

4. `contains` que retorna `true` si el elemento está en la lista y `false` en caso contrario.
5. Construya la función de soporte `void swap(lNode * n1, lNode *n2)` que itercambia dos nodos en la lista.

Tablas Hash

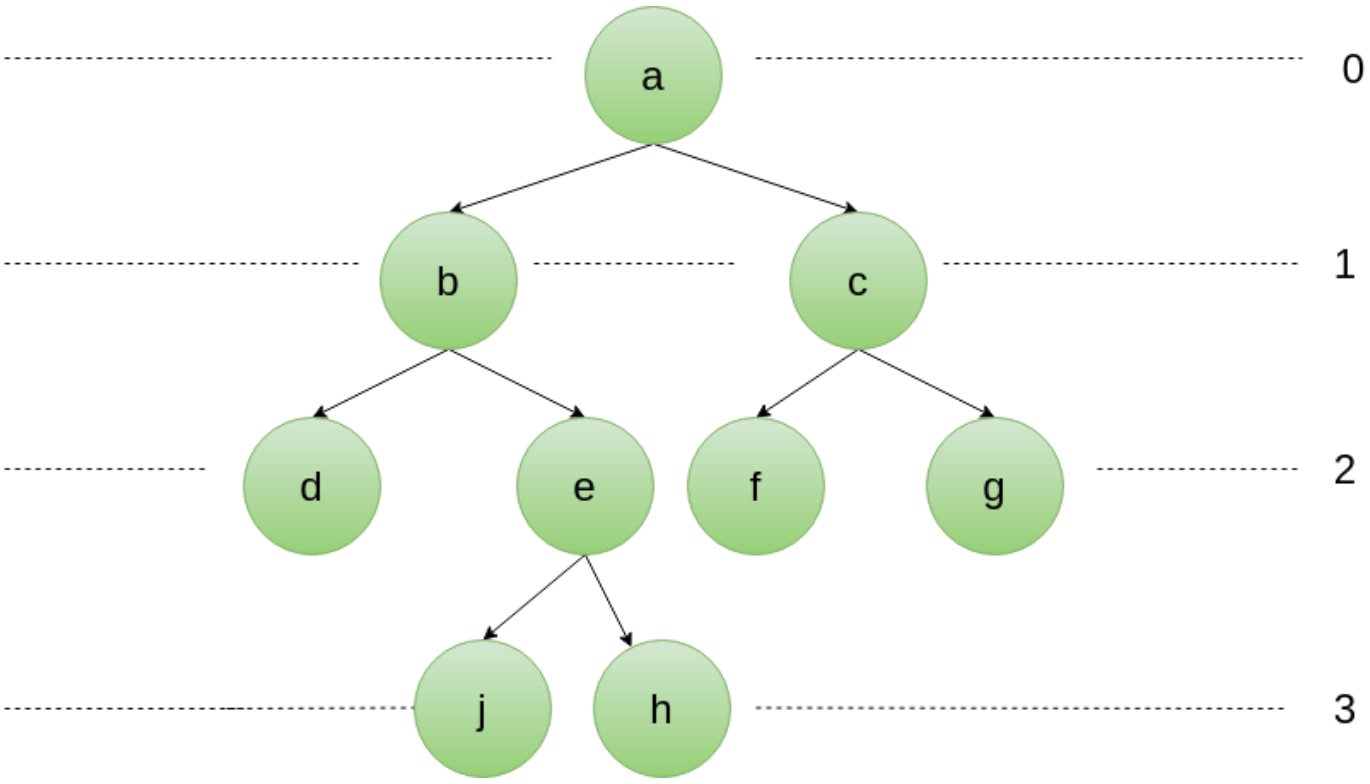
Dada la implementación de tablas hash mediante exploración de direcciones se solicita construir en **C++**.

> Recordatorio:
>
> 1. Método de plegamiento: Separar la clave en varias sub-claves.
> 2. Método de mitad del cuadrado: Elevar al cuadrado y seleccionar de forma determinística una cantidad de digitos menor al rango de la clave.
> 3. Dado una constante real **R** y una clave **k** y el tamaño de la tabla **m**.
>   1. Multiplicar **R*k**.
>   2. Calcular **d=Rk-parteEntera(Rk)** (o lo mismo que **d=parteDecimal(Rk)**).
>   4. Calcular **h(k)=parteEntera(m*d)**

1. La función `hash(key_t k)` mediante
    1. El método del plegamiento.
    2. El método de mitad del cuadrado.
    3. El método de la multiplicación.

Árboles

2. Dado el siguiente árbol:

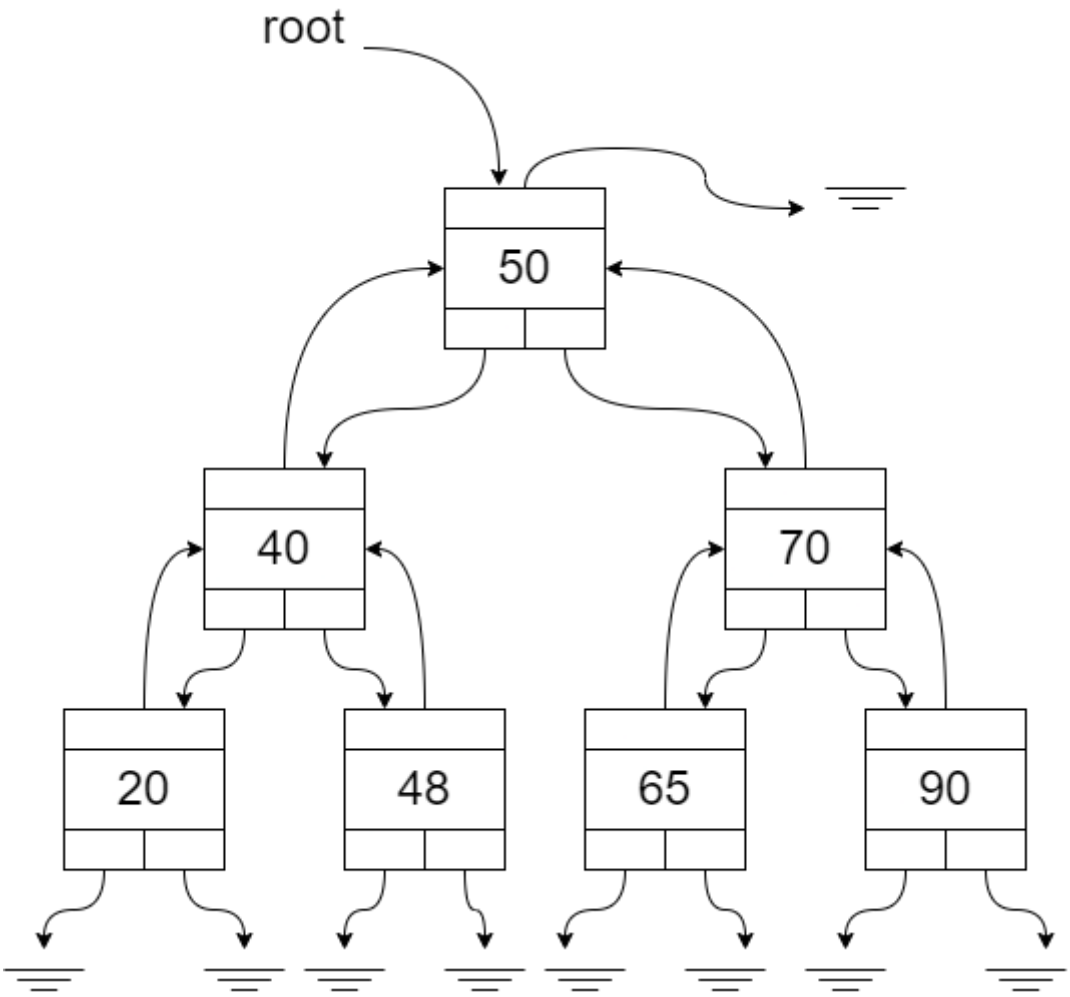


realice un recorrido:

1. Por niveles.
2. En Pre-Orden
3. En Post-Orden
4. En In-Orden

3. En un diagrama de nodos, muestre el resultado final de la operación *buildABB* que construye un árbol binario de búsqueda no balanceado. Esto, para la secuencia {4,8,2,3,9,1}. Una vez construido, realize la operación de eliminar la raiz en un diagrama de nodos.

4. Dado el siguiente arbol binarios de búsqueda:



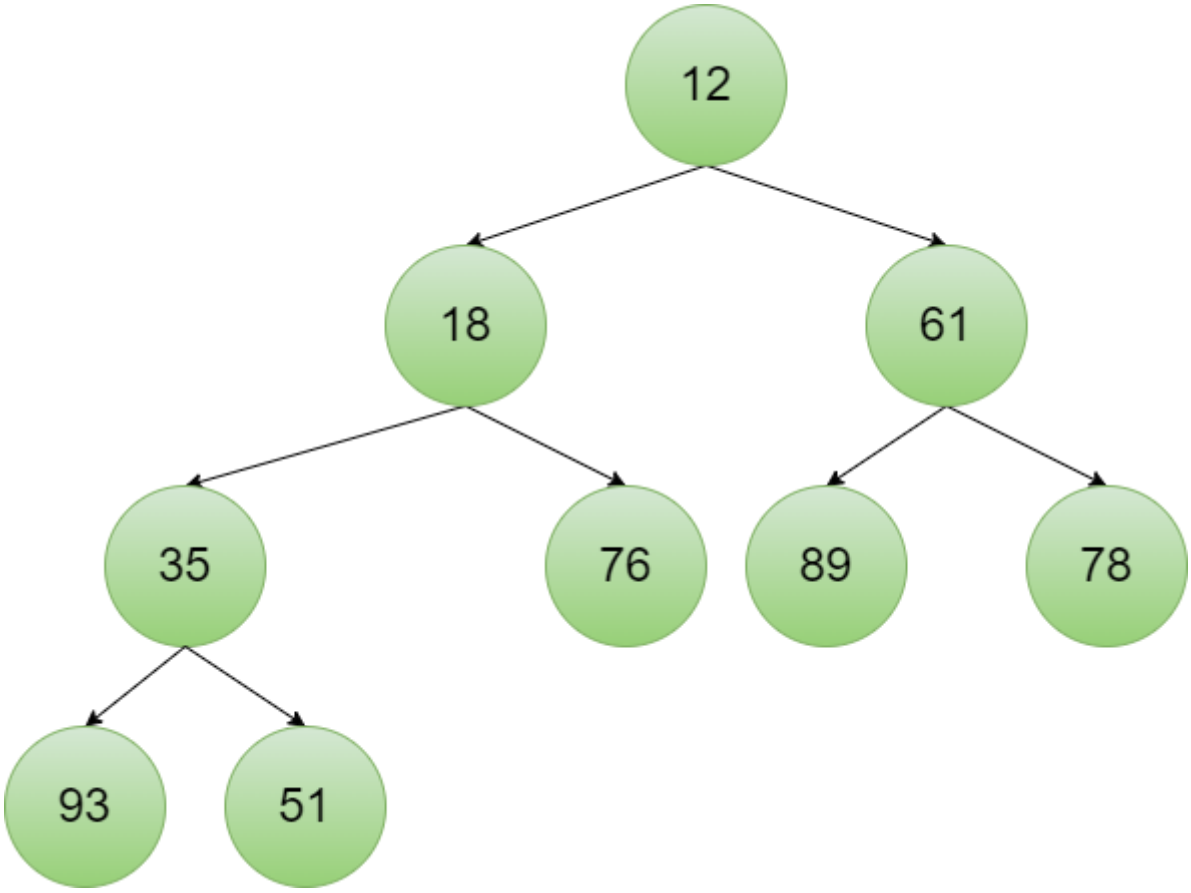
Muestre el resultado de eliminar el nodo que contiene al 70.

5. Para las siguiente secuencias:

1. {2,7,4,0,3,8,5,1,9}
2. {26,74,38,56,75,34,76,34,56,82}
3. {b,a,e,u,h,l,q,f,l,i}

En un diagrama de nodos, muestre el resultado final de la operación *buildMinHeap* que construye un minHeap. También realice esto para un MaxHeap.

6. Muestre paso a paso, en un diagrama de nodos, el resultado de la operación remove para este heap:



7. Muestre paso a paso, en un diagrama de nodos, el resultado de la operación remove para este heap: