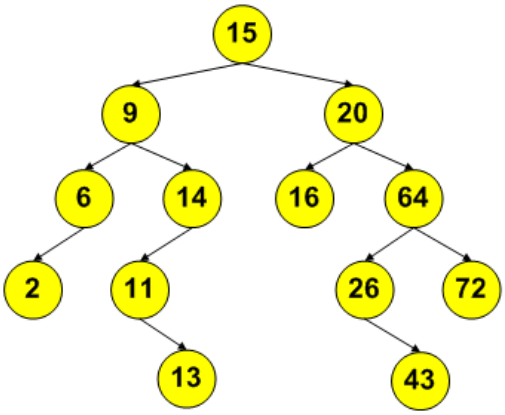
Tema: Introducción a Árboles

Apellido y Nombre: ..... Fecha:...../...../.....

EJEMPLOS

Operaciones sobre árboles binarios: Dado el siguiente árbol binario de búsqueda



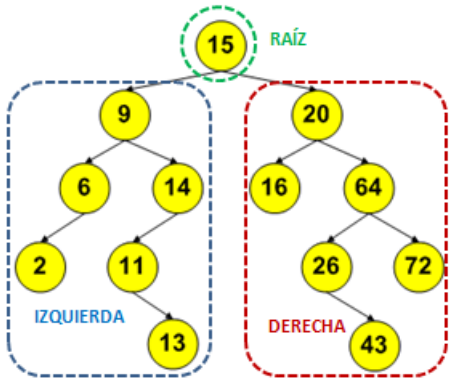
a) Defina la estructura de datos correspondiente
b) Muestre el recorrido en pos-orden del árbol
c) Diseñe un algoritmo recursivo que determine cuántos nodos del árbol tienen 2 descendientes
d) Elimine los nodos 15, 20 y 16 (aplique criterio mayor de menores cuando sea necesario)

Definición de la estructura

```
typedef struct tnodo *pnodo;
typedef struct tnodo { int dato;
                       pnodo izq;
                       pnodo der;
                     }
```

Recorrido en Pos-Orden

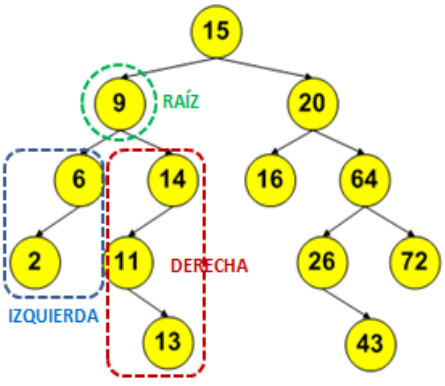
El recorrido en pos-orden de un árbol binario, básicamente, consiste en recorrer la estructura del árbol procesando primero el subárbol izquierdo, luego el subárbol derecho y finalmente el nodo raíz. Esto se aplica a cada subárbol que compone el árbol original.



En el árbol binario de la figura se identificaron el subárbol izquierdo, el subárbol derecho y el nodo raíz. Aplicando el algoritmo de pos-orden, se procesarán

SUBÁRBOL IZQUIERDO, SUBÁRBOL DERECHO, 15 (RAÍZ)

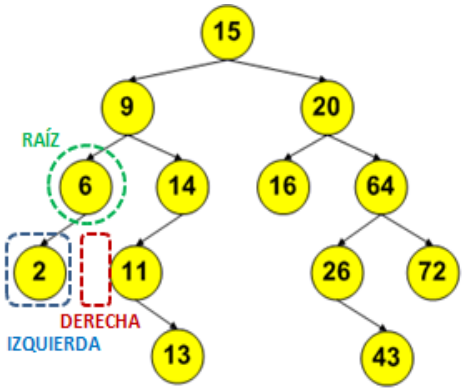
Para realizar el recorrido completo es necesario procesar todos los elementos del subárbol izquierdo, luego todos los elementos del subárbol derecho y por último la raíz. Por tanto, el algoritmo de recorrido debe aplicarse nuevamente a cada subárbol.



Al procesar el árbol de raíz 9 (subárbol izquierdo del original) el recorrido será

SUBÁRBOL IZQUIERDO, SUBÁRBOL DERECHO, 9 (RAÍZ)

Ahora deben procesarse los subárboles izquierdo y derecho del nodo 9. Nuevamente, se aplica la estrategia de recorrido.



Al procesar el árbol binario de raíz 6 el recorrido será:

SUBÁRBOL IZQUIERDO, SUBÁRBOL DERECHO, 6 (RAÍZ)

Luego, debe aplicarse el algoritmo de recorrido a los subárboles izquierdo y derecho del nodo 6.

Recorrido del subárbol izquierdo: **NULO, NULO, 2 (RAÍZ)**

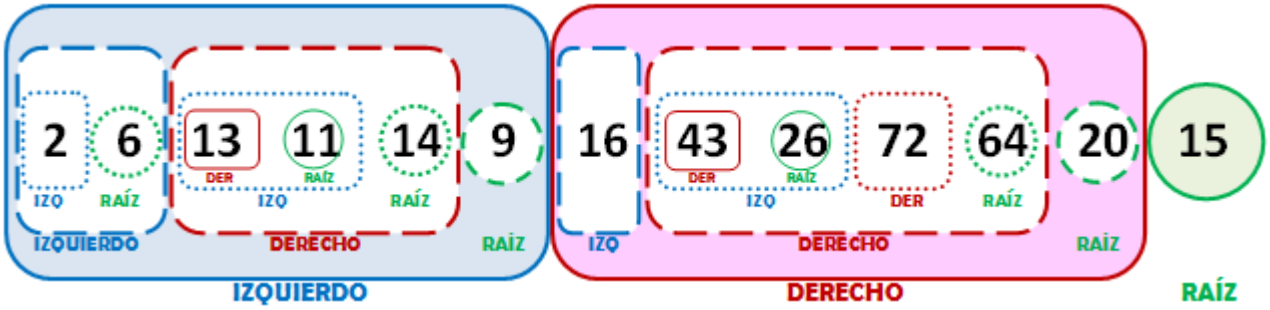
Recorrido del subárbol derecho: NULO

Esto se repite por cada subárbol.

Así, el recorrido en pos-orden del árbol binario procesará los nodos en el siguiente orden:

2, 6, 13, 11, 14, 9, 16, 43, 26, 72, 64, 20, 15

Esta secuencia se consigue al recorrer los nodos como se muestra en el siguiente esquema:



Nótese que el árbol original y cada subárbol se recorre de la misma forma: **IZQUIERDA**, **DERECHA**, **RAÍZ**.

Algoritmo recursivo

Para resolver el problema planteado se debe desarrollar un algoritmo recursivo que determine cuántos nodos del árbol binario poseen 2 descendientes. Para ello, el algoritmo propuesto debe recorrer el árbol verificando, por cada nodo visitado, la existencia de 2 descendientes. Al tratarse de un algoritmo recursivo las instrucciones se organizarán en 2 partes: *caso base* y *regla recursiva de construcción*. El *caso base* se asociará al árbol vacío y su solución será cero (no existen nodos que contar). Mientras que la *regla de recursiva de construcción* se asociará a la evaluación del nodo actual (contar o no el nodo) y los llamados recursivos (subárboles izquierdo y derecho). En el código se emplea una variable auxiliar *c* para contar los nodos con 2 descendientes (punteros *izq* y *der* no nulos). Nótese que el llamado recursivo se realiza por cada nodo visitado, tenga o no la cantidad de descendientes buscados (de otra forma el recorrido se detendrá en el primer nodo que no cumpla la condición).

```
int contar_nodos2(pnodo a)
{ int c;
  if (a==NULL)
    return 0;
  else
  { if (a->izq!=NULL && a->der!=NULL)
      c=1;
    else
      c=0;
    return contar_nodos2(a->izq)+contar_nodos2(a->der)+c;
  }
}
```

Caso Base (Árbol vacío)

c=1 cuando el nodo tiene 2 descendientes

c=0 cuando el nodo tiene 0 o 1 descendientes

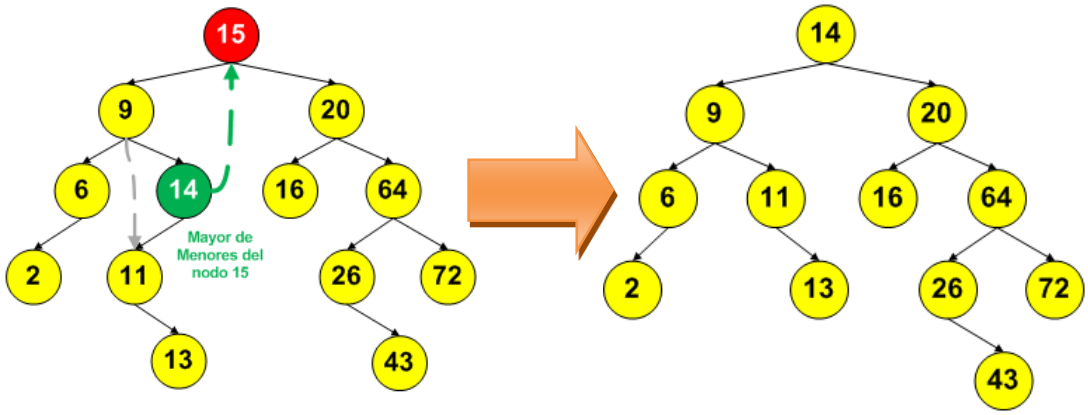
Regla Recursiva de Construcción (Evaluación del nodo actual y Llamado recursivo)

Eliminación de nodos del árbol binario

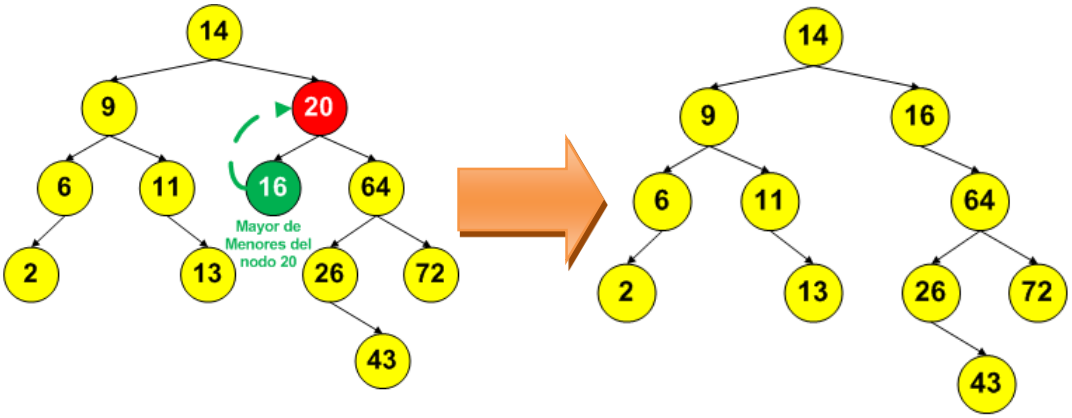
Al eliminar un nodo lo primero que debe identificarse es el caso de eliminación a aplicar: *nodo hoja*, *nodo con 1 descendiente* o *nodo con 2 descendientes*. Si el nodo es una hoja se elimina directamente mientras que si posee 1 descendiente entonces este

descendiente debe reemplazar al nodo eliminado. Sin embargo, cuando el nodo posee 2 descendientes es necesario utilizar un criterio para elegir el nodo que ocupará el lugar del eliminado (Mayor de Menores o Menor de Mayores).

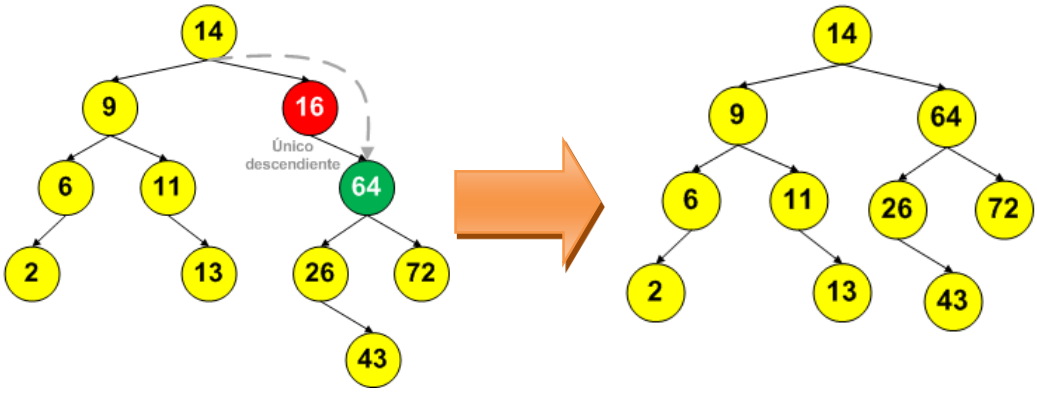
Eliminación del nodo 15: al tratarse de un nodo con 2 descendientes debe aplicarse un criterio para seleccionar el nodo de reemplazo (el enunciado indica Mayor de Menores). En este caso, el nodo 14 es el mayor de menores para el nodo 15 (de los valores que se encuentran a la izquierda de 15, 14 es el que está más a la derecha). Además, al reubicar el nodo 14, el nodo 11 (único descendiente de 14) pasa a ocupar su lugar en el árbol (aquí se aplicó el 2do caso de eliminación).



Eliminación del nodo 20: el nodo 20 posee 2 descendientes por lo que debe aplicarse un criterio para seleccionar el nodo de reemplazo (Mayor de Menores según indica el enunciado). En este caso, el nodo 16 sustituirá al 20 (mayor valor del lado izquierdo de 20). No es necesario reubicar otros nodos ya que 16 no tenía descendientes.



Eliminación del nodo 16: el nodo 16 sólo posee un descendiente por lo que es evidente cuál será el nodo que lo reemplazará. En este caso, el nodo con valor 64 (único descendiente de 16) reemplazará al nodo 16.

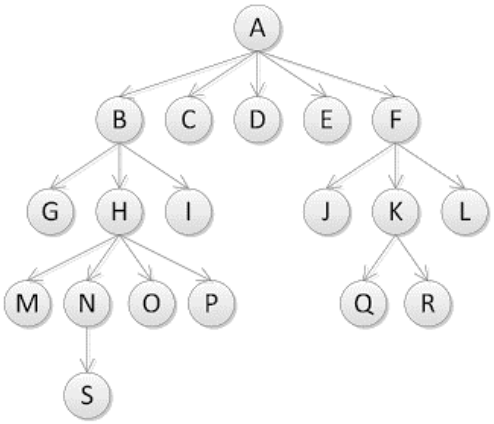


EJERCICIOS

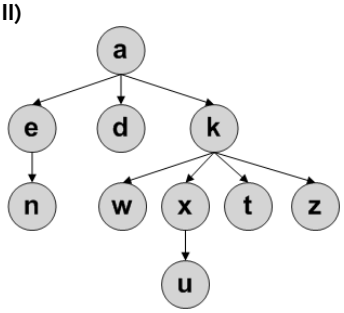
1) Dados los siguientes árboles, determine para cada uno de ellos:

   a) niveles del árbol
   b) altura del árbol
   c) peso del árbol
   d) grado del árbol
   e) nodo raíz, nodos interiores y nodos terminales
   f) nodos ascendientes y descendientes de cada nodo
   g) representación parentizada

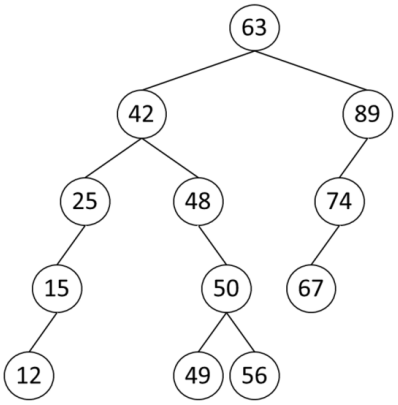
I)



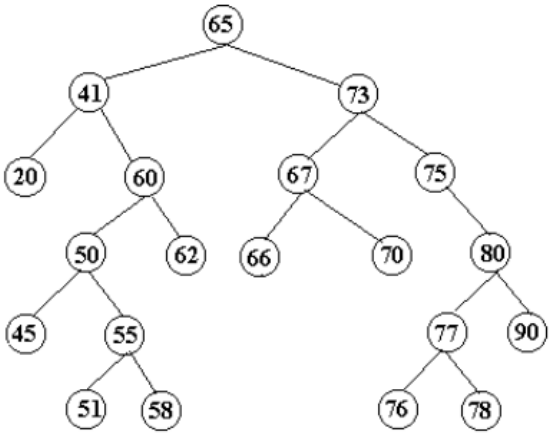
II)



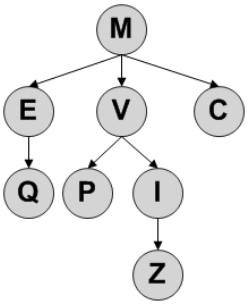
III)



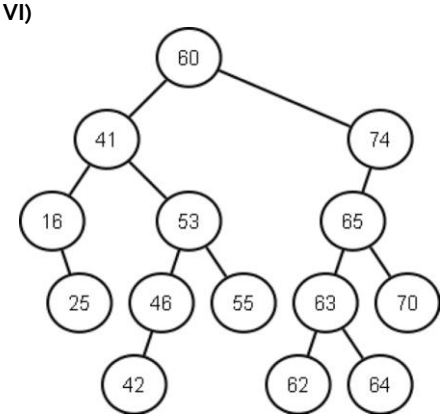
IV)



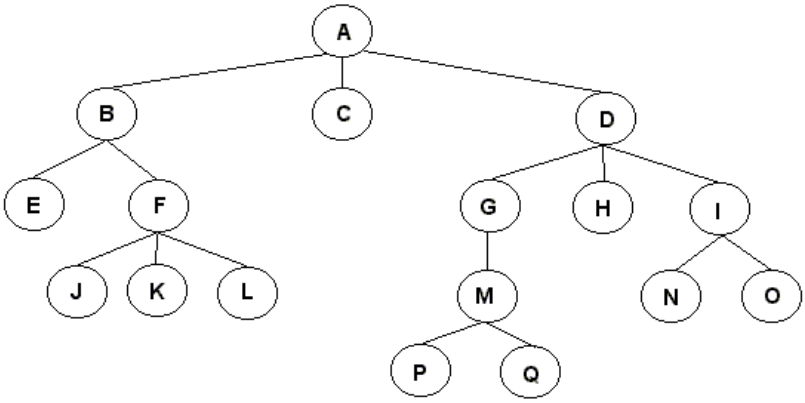
V)



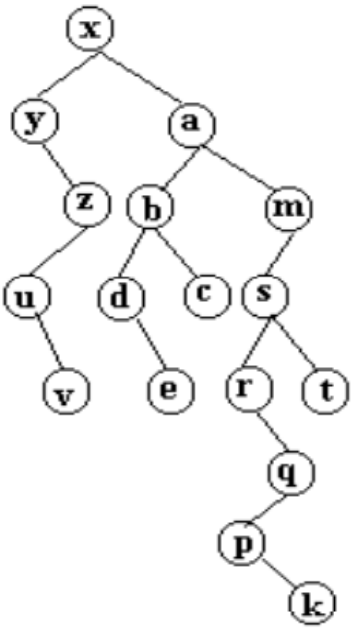
VI)



VII)

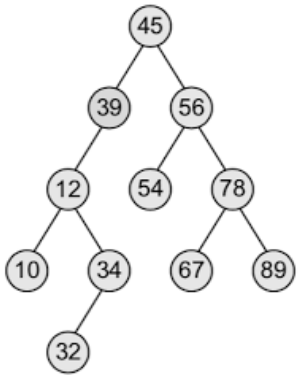


VIII)

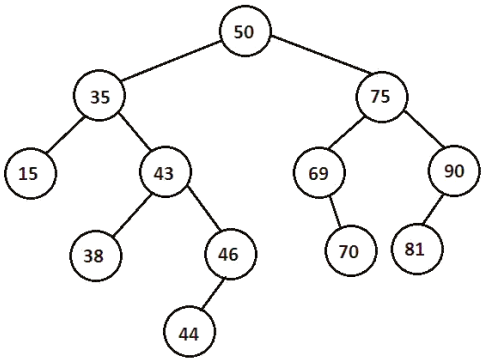


2) Dados los siguientes árboles binarios realice el recorrido en pre-orden, en-orden y pos-orden para cada uno.

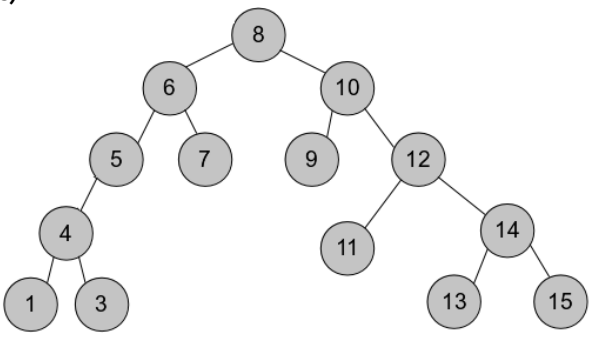
a)



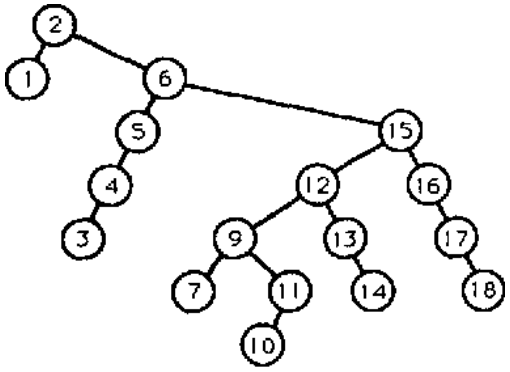
b)



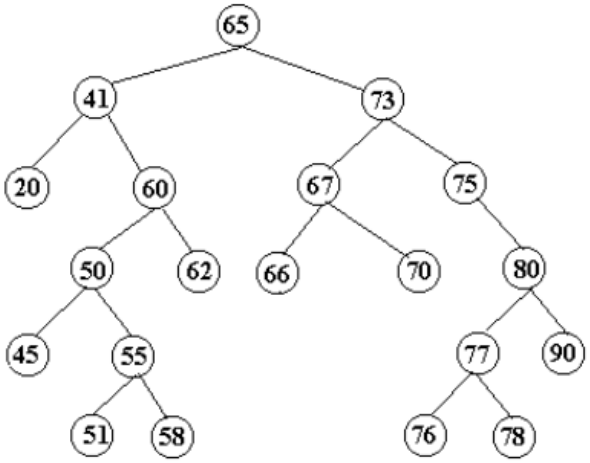
c)



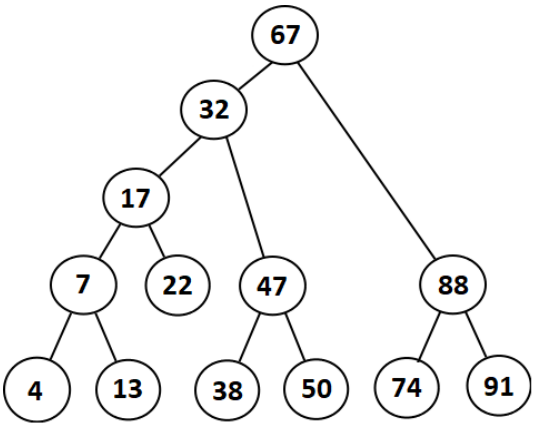
d)



e)



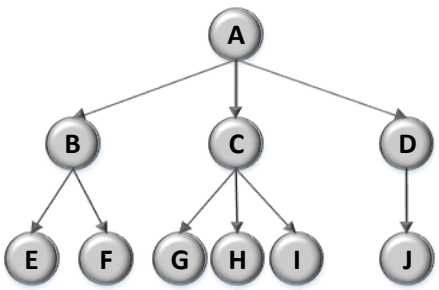
e)



3) Considerando un árbol binario, diseñe los algoritmos recursivos que permitan determinar:

a) cantidad de nodos del árbol.
b) cantidad de hojas del árbol.
c) suma de todos nodos del árbol.
d) altura del árbol.
e) si se trata de un árbol binario completo o no.
f) si dos árboles binarios son similares
g) si dos árboles binarios son equivalentes
h) cantidad de nodos del árbol con 1 descendiente.

4) Dado el siguiente árbol binario:



a) defina la estructura de datos correspondiente,
b) muestre el contenido del árbol considerando que se recorrerá en *pre-orden*,
c) obtenga la representación parentizada,
d) codifique un algoritmo que cuenta las hojas del árbol.
e) codifique un algoritmo que determine si un valor pertenece o no al árbol.

5) Dadas las siguientes secuencias de datos, genere los correspondientes árboles binarios de búsqueda.

   a) 9, 5, 6, 20, 4, 1, 17, 7, 2, 15
   b) p, s, d, f, s, t, g, w, o, z
   c) 12, 43, 16, 34, 7, 18, 15, 6, 21, 42
   d) F, N, Q, G, A, W, E, D, R, V
   e) 13, 28, 46, 22, 11, 31, 37, 24, 6, 48
   f) n, r, z, k, e, f, w, b, t, h, p

6) Considerando el algoritmo de **inserción** de árboles binarios de búsqueda

   a) modifíquelo de modo que no se permita la inserción de valores repetidos (los nodos no agregados deben liberarse). Aplique el algoritmo modificado a la secuencia 11, 7, 16, 5, 22, 20, 28, 8, 49, 32, 13, 17, 6, 12.
   b) modifíquelo de modo que el subárbol izquierdo del nodo *i* contenga los valores mayores a *i*, mientras que el subárbol derecho guarde los valores menores a *i*.
   c) aplicando el algoritmo del ítem b) dibuje el árbol binario de búsqueda correspondiente a la secuencia 18, 16, 54, 21, 33, 14, 17, 44, 11, 27

7) Considerando el árbol binario de búsqueda creado en el ítem 6.c), modifique el algoritmo de búsqueda y realice la prueba de escritorio para los valores *33, 13* y *27*. ¿Qué modificaciones realizó al algoritmo de búsqueda básico?

8) Considerando un árbol binario de búsqueda de caracteres, realice lo siguiente:

   a) defina la estructura de datos correspondiente,
   b) dibuje el árbol binario de búsqueda para la secuencia K, M, R, C, A, F, W, D, S, H, X, G, E, O
   c) desarrolle un algoritmo recursivo que cuente los nodos del árbol que contengan vocales. Suponga que el árbol sólo almacena mayúsculas.
   d) desarrolle un algoritmo recursivo que genere una copia del árbol.

9) Considerando un árbol binario de búsqueda de valores enteros, realice lo siguiente:

   a) defina la estructura de datos correspondiente,
   b) dibuje el árbol binario de búsqueda para la secuencia numérica 19, 47, 61, 73, 13, 61, 67, 7, 11, 97
   c) modifique el algoritmo de inserción de modo que sólo se agreguen valores primos. Considere que los nodos no agregados deben liberarse.
   d) desarrolle un algoritmo recursivo que muestre el contenido del árbol en orden decreciente.

10) Considerando un árbol binario de búsqueda de valores enteros, realice lo siguiente:

   a) defina la estructura de datos correspondiente,
   b) dibuje el árbol binario de búsqueda para la secuencia numérica 17, 24, 27, 16, 22, 29, 51, 43, 44, 25
   c) desarrolle un algoritmo recursivo que muestre en pre-orden, en-orden o pos-orden el contenido del árbol, según un parámetro de opción.
   d) desarrolle un algoritmo recursivo que cuente los valores capicúa, los valores primos y los valores primo-capicúa almacenados en árbol.

11) Dos árboles binarios que tienen la misma estructura, es decir, que presentan la misma forma se denominan árboles binarios similares. Considerando esto, desarrolle un algoritmo recursivo que calcule, nodo a nodo, el producto entre los valores de dichos árboles generando un tercer árbol con los resultados obtenidos. Considere que el producto se resuelve mediante sumas sucesivas.

12) Analice los siguientes fragmentos de código, describa las acciones que realizan y determine su propósito:

```
bool enigma (pnodo a, pnodo b)
{ if (a==NULL && b==NULL)
      return true;
  else
   { if (a!=NULL && b!=NULL && a->dato==b->dato)
       return enigma(a->izq,b->izq) && enigma(a->der,b->der);
     else
       return false; }
}
```

```
bool misterio (pnodo a)
{ if (a!=NULL)
  { if (a->izq!=NULL && a->der!=NULL)
     {
        return misterio(a->izq) && misterio(a->der);
     }
    else
    {  if (a->izq==NULL && a->der==NULL)
         return true;
      else
         return false;
    }
  }
  else
    return false;
}
```

13) Analice el siguiente fragmento de código, describa las acciones que realiza y determine su propósito. ¿Cuál es el comportamiento del algoritmo si en lugar de utilizar una cola se emplea una pila?

```
void secreto (pnodo a)
{ pnodo aux;
  tcola q;
    if (a==NULL)
     cout << "VACIO" << endl;
    else
     { iniciar_cola(q);
       agregar_cola(q,a);
       while (!cola_vacia(q))
       { aux=quitar_cola(q);
         cout << aux->dato << endl;
         if (aux->der!=NULL);
           agregar_cola(q,aux->der);
         if (aux->izq!=NULL)
           agregar_cola(q,aux->izq);
       }
     }
}
```

Definición del TDA árbol

```
typedef struct tnodo *pnodo;
typedef struct tnodo {
                       int dato;
                       pnodo izq;
                       pnodo der;
                      }
```

Definición del TDA cola utilizado

```
const int MAX=100,
typedef pnodo punteros[MAX];
typedef struct tcola {
                       punteros datos;
                       int frente;
                       int final;
                      }
```

14) Analice los siguientes fragmentos de código, describa las acciones que realizan y determine su propósito:

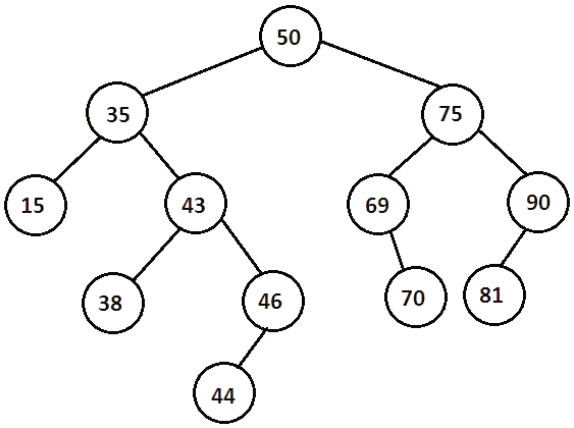
a)

```
bool misterio (pnodo a, int b)
{ pnodo aux;
  bool e=false;
  if (a!=NULL)
  { aux=a;
    while (aux!=NULL && e==false)
    { if (aux->dato==b);
        e=true;
      else
       { if (b < aux->dato)
           aux=aux->izq;
         else
           aux=aux->der;
       }
    }
  }
  return e;
}
```

b)

```
void incognita (pnodo &a, pnodo b)
{ pnodo aux1,aux2;
  if (a==NULL)
    a=b;
  else
    { aux1=a;
      while (aux1!=NULL)
      { aux2=aux1;
        if (aux1->dato > b->dato)
          aux1=aux1->izq;
        else
          aux1=aux1->der;
      }
      if (aux2->dato > b->dato)
         aux2->izq=b;
      else
         aux2->der=b;
    }
}
```

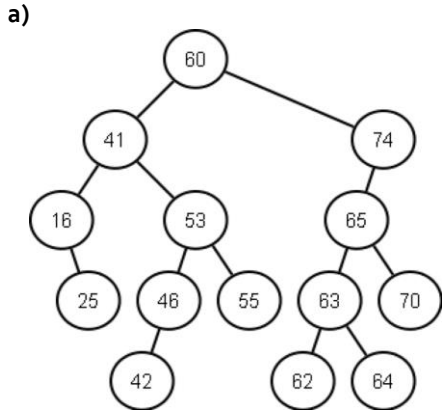
15) Dado el siguiente árbol binario:



a) defina la estructura de datos correspondiente,
b) muestre el recorrido DERECHA-IZQUIERDA-RAIZ del árbol
c) codifique un algoritmo recursivo que muestre los valores del árbol mayores a un valor especificado.
d) codifique un algoritmo recursivo que determine si el árbol contiene exclusivamente valores impares o no.

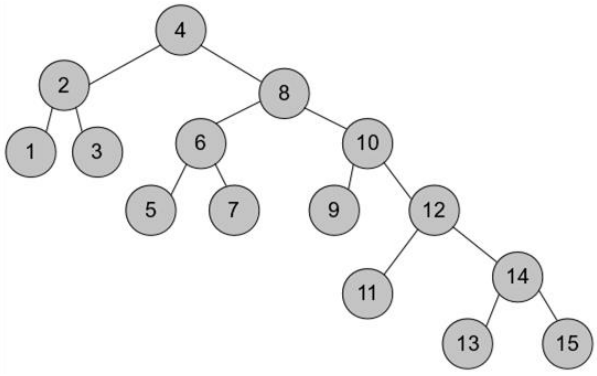
16) Dados los siguientes árboles binarios de búsqueda, aplique **gráficamente** el algoritmo de eliminación para los nodos que se especifican. De ser necesario, utilice para la eliminación el criterio especificado en cada caso.

a)



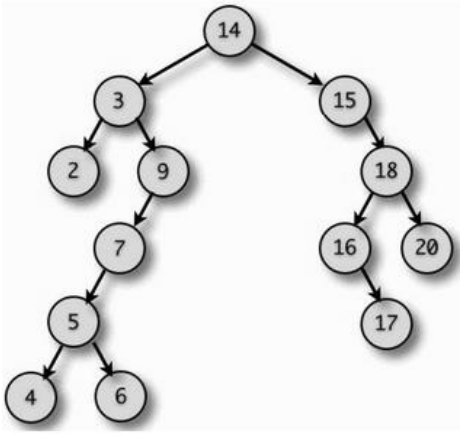
Eliminar 62, 60 y 41
si fuera necesario MENOR DE MAYORES

b)



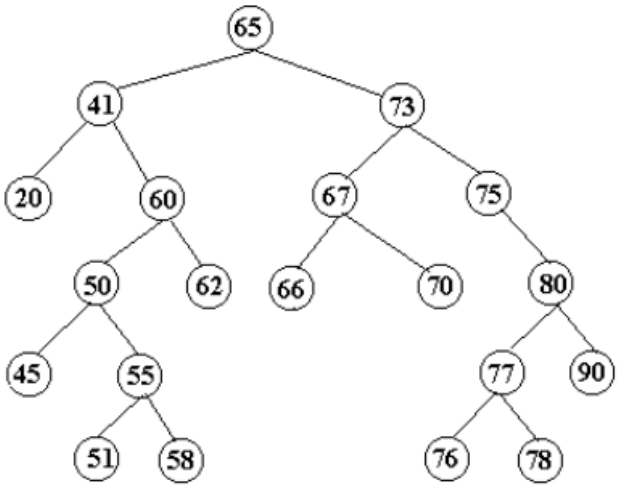
Eliminar 9, 10 y 8
si fuera necesario MAYOR DE MENORES

c)



Eliminar 15, 5 y 3
si fuera necesario MENOR DE MAYORES

d)

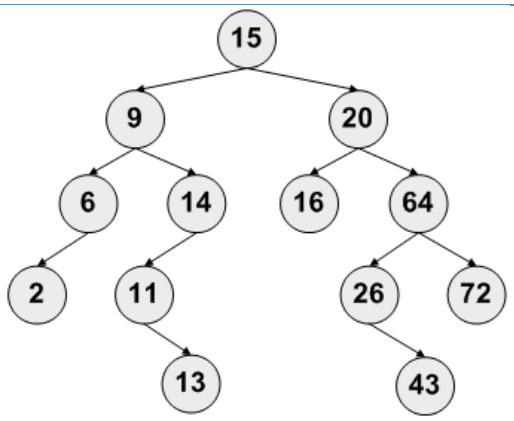


Eliminar 50, 65 y 75
si fuera necesario MAYOR DE MENORES

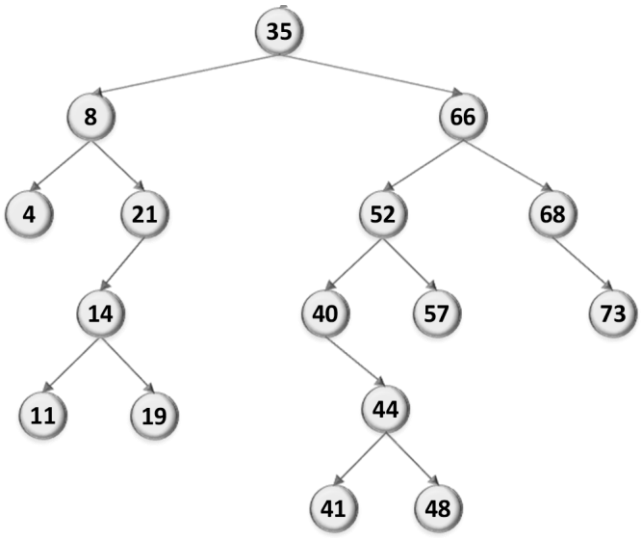
17) Dados los siguientes árboles binarios de búsqueda, aplique el algoritmo de eliminación para los nodos que se especifican. De ser necesario, utilice para la eliminación el criterio MENOR DE MAYORES.

a) 44(20(12,30(24)),49(52)) Eliminar 20 y 44
b) 16(10(4(1),14(12,15)),47(21,61)) Eliminar 14 y 10
c) 41(30(29,36(34(33(32),35))),43(42,46(44(45),48))) Eliminar 41 y 43

18) Dado el siguiente árbol binario de búsqueda elimine los nodos 20, 9 y 15, dibujando tras cada eliminación el árbol resultante. Cuando sea necesario aplique el criterio MENOR DE MAYORES.



19) Dado el siguiente árbol binario de búsqueda elimine los nodos 57, 8 y 66, dibujando tras cada eliminación el árbol resultante. Cuando sea necesario aplique el criterio MAYOR DE MENORES.



20) Dado el siguiente árbol binario de búsqueda elimine los nodos 340, 151 y 268, dibujando tras cada eliminación el árbol resultante. Cuando sea necesario aplique el criterio MAYOR DE MENORES.