# ***Examen PED diciembre 2007***
# ***Modalidad 0***

**Normas:**
- La entrega del test no corre convocatoria.
- Tiempo para efectuar el test: **15 minutos**.
- Una pregunta mal contestada elimina una correcta.
- Las soluciones al examen se dejarán en el campus virtual.
- **Una vez empezado el examen no se puede salir del aula hasta finalizarlo.**
- En la **hoja de contestaciones** el verdadero se corresponderá con la **A**, y el falso con la **B**.

| | V | F | | |
|---|---|---|---|---|
| Las ecuaciones (vistas en clase) que permiten realizar la suma de números naturales son las siguientes:<br>VAR x, y: natural;<br><br>suma(x, cero) = x<br>suma(cero, x) = x<br>suma(x, suc(y)) = suma(suc(x), y) | ❑ | ❑ | 1 | F |
| En C++, la siguiente declaración es INCORRECTA : const int& a = 1; | ❑ | ❑ | 2 | F |
| Dadas las clases TDir y TVectorDir con todos sus métodos implementados:<br>constructor, constructor sobrecargado, sobrecarga del corchete, sobrecarga del operador salida (se muestra el contenido de cada posición del vector dejando un espacio), etc.<br>Nota: El constructor por defecto crea un vector vacío.<br>El constructor a partir de una dimensión, pone todos los elementos a 0.<br>*class TDir* *class TVectorDir* *main()*<br>*{* *{* *{*<br>*public: ….* *public: ….* *TDir a(1,1);*<br>*private:* *private:* *TVectorDir v;*<br>*int e1;* *TDir *vector;* *cout<<"v_antes="<<v<<endl;*<br>*int e2;* *int longitud;* *v[1]=a;*<br>*};* *};* *cout<<"v_despues="<<v<<endl;*<br>*}*<br>El resultado obtenido tras la ejecución del *main()* sería:<br>*v_antes= 0 0*<br>*v_despues= 1 1* | ❑ | ❑ | 3 | F |
| La complejidad espacial es la cantidad de recursos espaciales que un algoritmo consume o necesita para su ejecución. | ❑ | ❑ | 4 | V |
| La complejidad temporal de un algoritmo depende de la complejidad espacial del mismo. | ❑ | ❑ | 5 | F |
| La semántica de la operación *desencolar* vista en clase es la siguiente:<br>VAR c: cola, x: item;<br>si esvacia( c ) entonces<br>desencolar( encolar( c, x ) ) = crear_cola ( )<br>si no desencolar( encolar( c, x ) ) = encolar( desencolar ( c ), x ) | ❑ | ❑ | 6 | V |
| Un árbol con un único nodo es un árbol lleno. | ❑ | ❑ | 7 | V |
| Un árbol con un único nodo tiene un único camino cuya longitud es 1. | ❑ | ❑ | 8 | F |
| Dados los recorridos de preorden, postorden y niveles de un árbol binario de altura 7 y 64 hojas es posible reconstruir un único árbol binario. | ❑ | ❑ | 9 | V |
| En la representación de conjuntos mediante listas, la complejidad espacial es proporcional al tamaño del conjunto representado. | ❑ | ❑ | 10 | V |
| En un grafo dirigido pueden existir infinitas aristas para un número “n” de vértices. | ❑ | ❑ | 11 | F |

# *Examen PED diciembre 2007*

**Normas:**

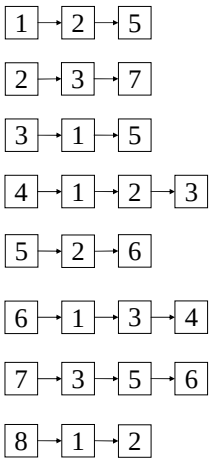
- Tiempo para efectuar el ejercicio: **2 horas**
- En la cabecera de cada hoja **Y EN ESTE ORDEN** hay que poner: ***APELLIDOS, NOMBRE***.
- Cada pregunta se escribirá en hojas diferentes.
- Se dispone de 20 minutos para abandonar el examen sin que corra convocatoria.
- Las soluciones al examen se dejarán en el campus virtual.
- Se puede escribir el examen con lápiz, siempre que sea legible
- **Todas las preguntas tienen el mismo valor.** Este examen vale el 60% de la nota de teoría.
- **Publicación notas**: se publicará un anuncio en el campus virtual.
- **Los alumnos que estén en 5ª o 6ª convocatoria deben indicarlo en la cabecera de todas las hojas**

**1.** Define la sintaxis y semántica de la operación **posición** que actúa sobre un vector y devuelve la posición menor sobre la que se ha asignado el valor que recibe como parámetro. Si no se ha asignado el valor en el vector se debe devolver 0. Ejemplo:

posición(asig(crear(), 3, b), a) = 0
posición(asig(asig(crear(), 3, b), 1, b), b) = 1
posición(asig(asig(asig(crear(), 3, b), c, 2), 1, b), b) = 1
posición(asig(asig(asig(crear(), 1, b), c, 2), 3, b), b) = 1

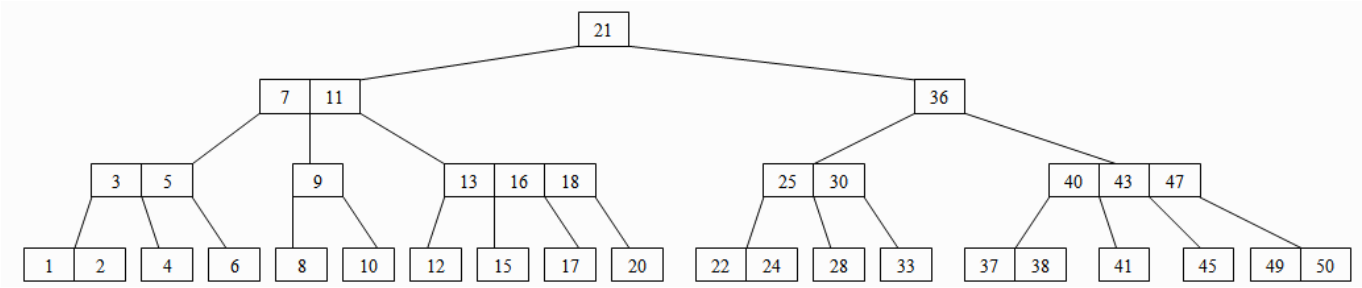
**2.** Dado el siguiente grafo no dirigido representado por la lista de adyacencia que se muestra a continuación:

a) Realiza el recorrido DFS(1), recorriendo la adyacencia de menor a mayor y obtén el bosque extendido en profundidad.
b) Calcula las componentes fuertemente conexas del grafo inicial.

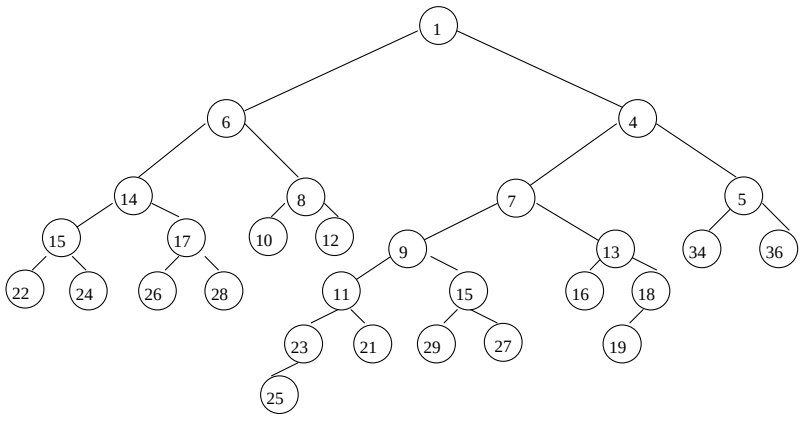



**3.** Sobre el siguiente árbol 2-3-4, realizar los siguientes borrados consecutivos: 25, 11, 21. Criterios:

a) Sustituir por el mayor de la izquierda
b) Si existen dos nodos adyacentes escoger 'r' hermano de la derecha




**4.** Dados el siguiente árbol leftist mínimo, realizar dos borrados sucesivos, y sobre el resultado final, la inserción de la clave 20.



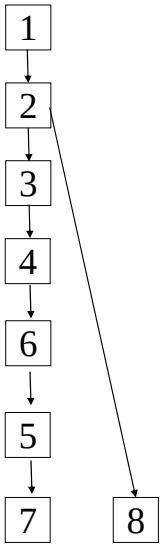
# *Examen PED diciembre 2007. Soluciones*

**1.**

```
posicion(vector, item) → entero
posicionAux(vector, item, entero) → entero
Var v:vector; i: entero; x: item;
posicion(crear(), x) = 0
si (x<>y) entonces
        posicion(asig(v, i, x), y) = posicion(v, y)
si no posicion(asig(v, i, x), y) = posicionAux(v, y, i)
posicionAux(crear(), x, i) = i
si (x<>y) entonces
        posicionAux(asig(v, i, x), y, j) = posicionAux(v, y, j)
si no   si (i < j) entonces
                posicionAux(asig(v, i, x), y, j) = posicionAux(v, y, i)
        si no  posicionAux(asig(v, i, x), y, j) = posicionAux(v, y, j)
```

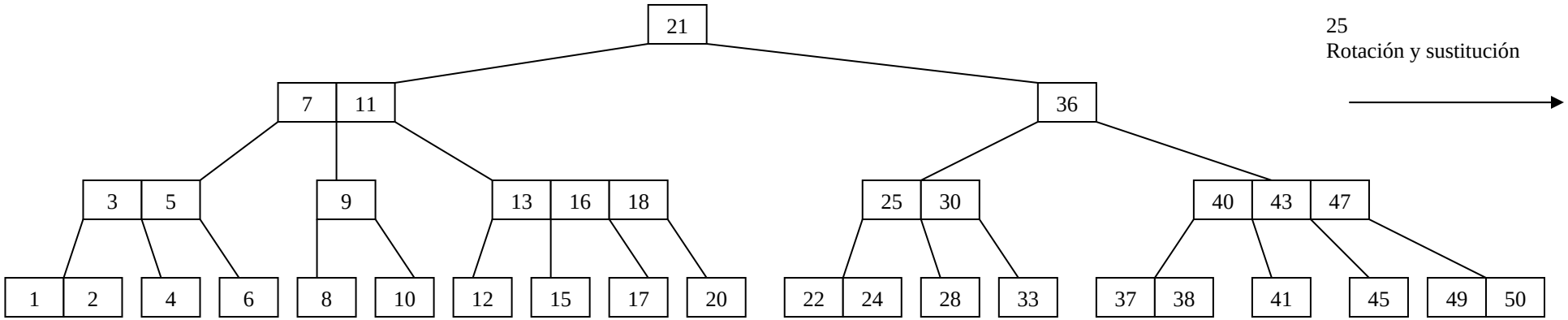
**2.**

a) DFS(1): 1,2,3,4,6,5,7,8

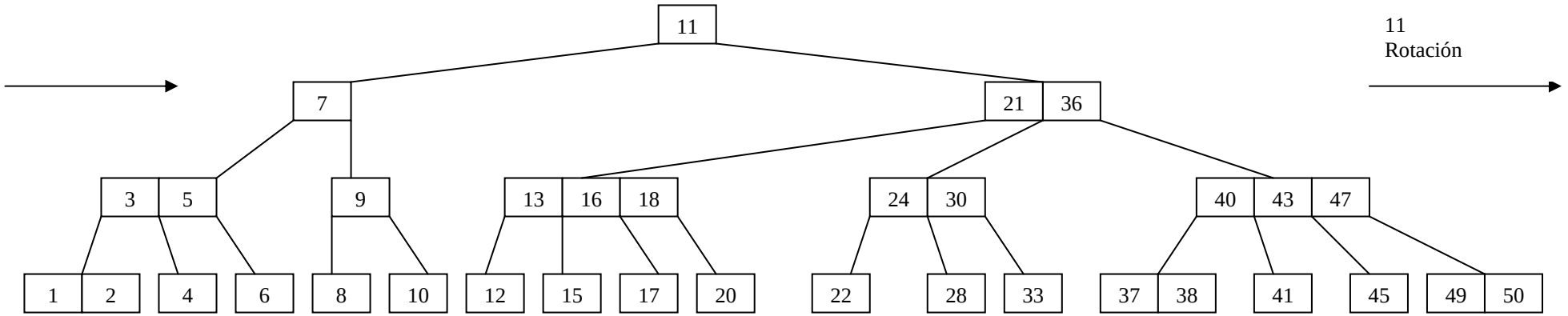



b) Componentes fuertemente conexas: {1,2,3,4,5,6,7,8}

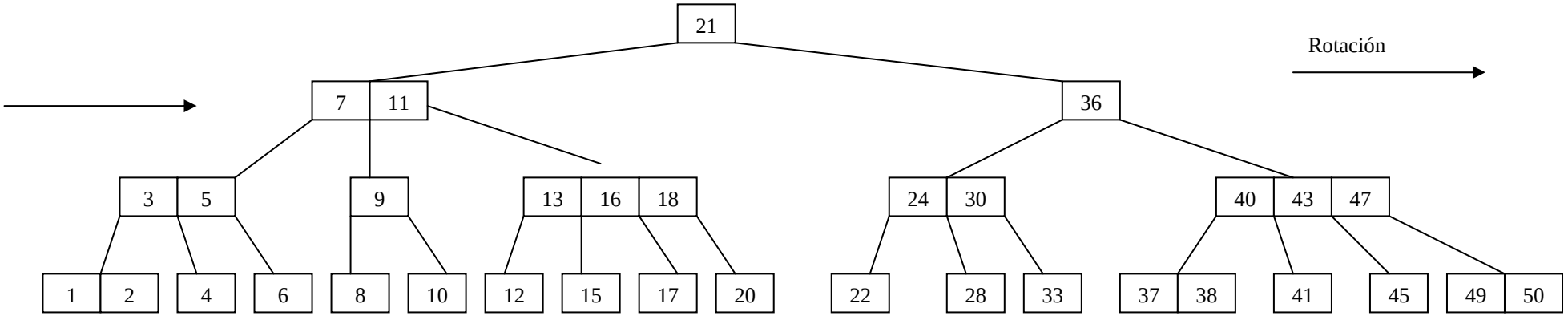
**3.**
Solución:

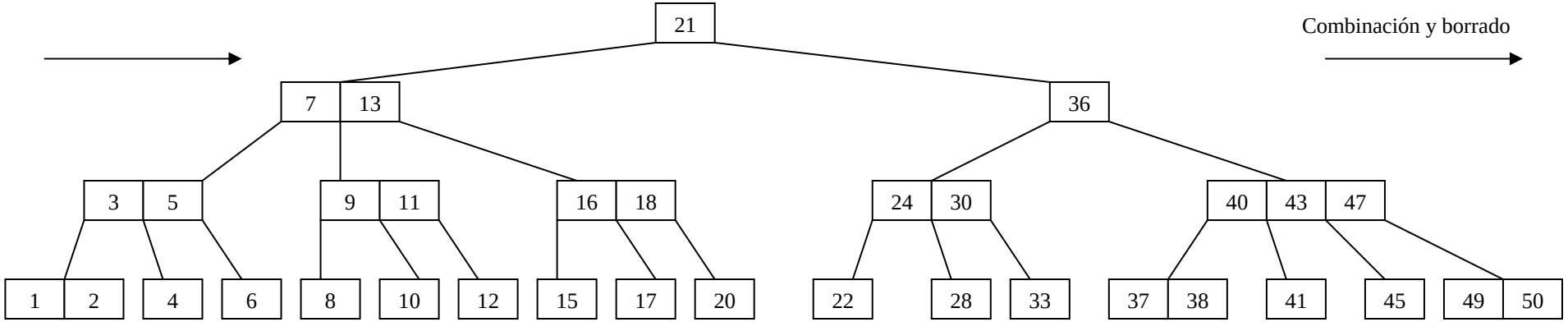


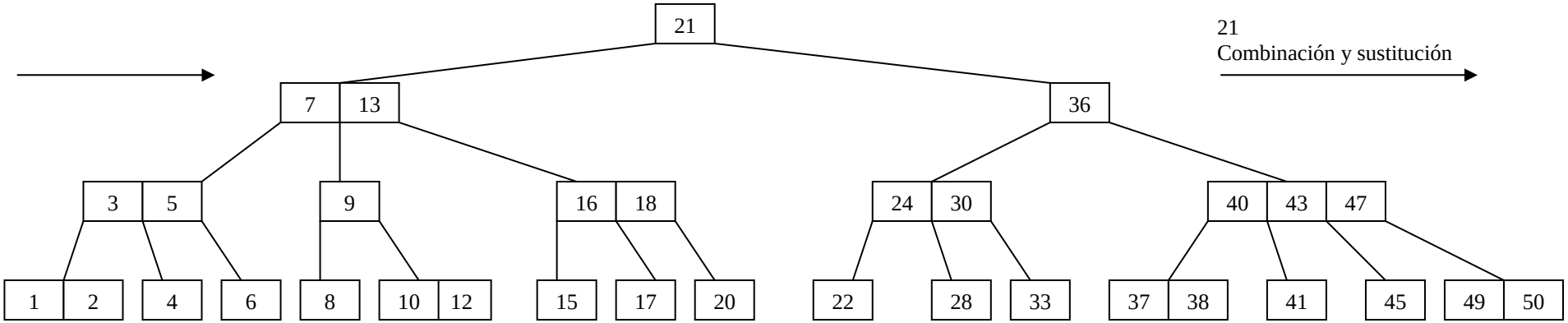

25
Rotación y sustitución

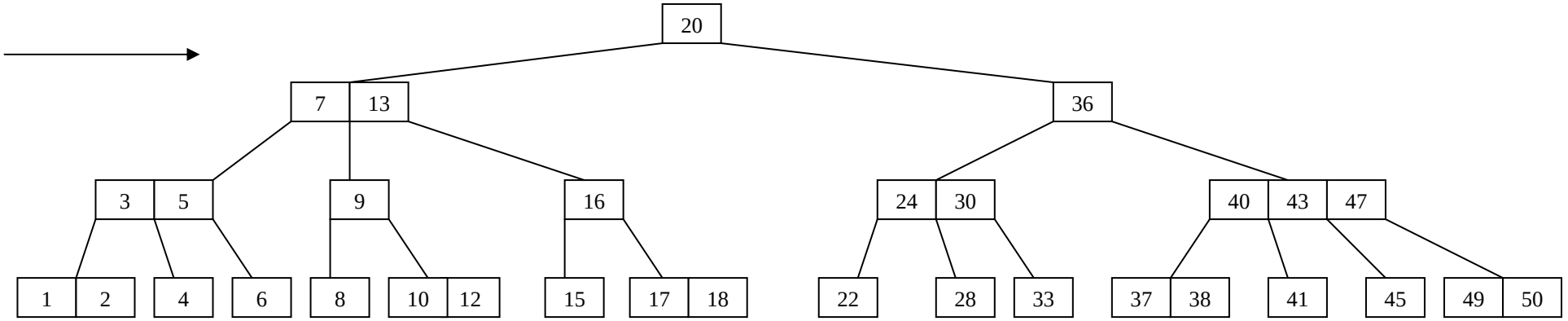



11
Rotación

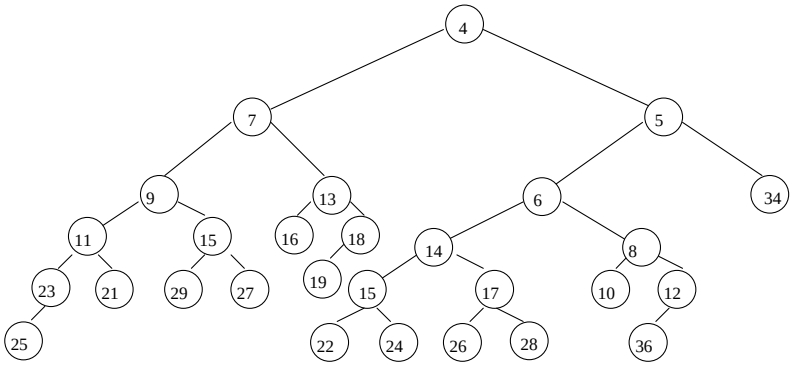

21
7
11
36
3
5
9
13
16
18
24
30
40
43
47
1
2
4
6
8
10
12
15
17
20
22
28
33
37
38
41
45
49
50
Rotación


21
7
13
36
3
5
9
11
16
18
24
30
40
43
47
1
2
4
6
8
10
12
15
17
20
22
28
33
37
38
41
45
49
50
Combinación y borrado

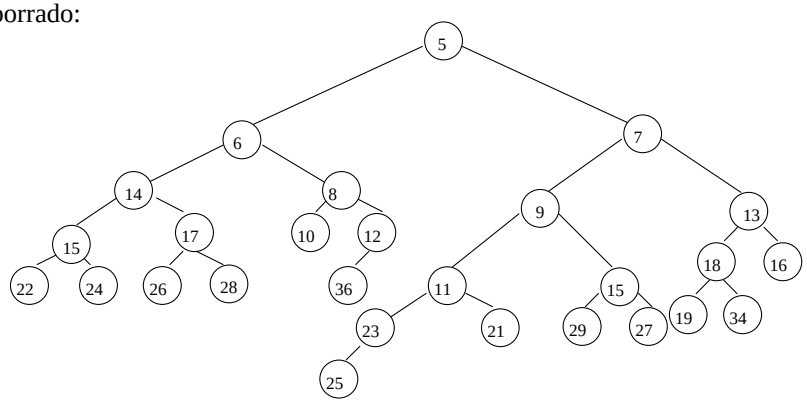

21
7
13
36
3
5
9
16
18
24
30
40
43
47
1
2
4
6
8
10
12
15
17
20
22
28
33
37
38
41
45
49
50
21
Combinación y sustitución




**4.** Primer borrado:



Segundo borrado:




Inserción del 20: