EJERCICIOS REALIZADOS EN CLASE – SOBRE ARBOLES

Programa que muestra los nodos de un árbol mediante recorrido en anchura mediante una cola

```
import java.util.Queue;
import java.util.LinkedList;
public class Ejemplo1 {

    static int tam=5;
    static int M[][]=new int[tam][tam];
    public static void main(String arg[]){

        M[2][0]=1;
        M[2][1]=1;
        M[1][3]=1;
        M[1][4]=1;

        prof(2);
    }
    static void prof(int raiz){
        Queue<Integer>c=new LinkedList<Integer>();
        c.add(raiz);
        while(!c.isEmpty()){
            int x=c.remove();
            System.out.print(x+" ");
            for(int i=0;i<=4;i++){
                if(M[x][i]==1)
                    c.add(i);
            }
        }
    }
}
```

Programa que muestra las hojas de un árbol mediante un recorrido en anchura

```
import java.util.Queue;
import java.util.LinkedList;
public class Ejemplo1 {

    static int tam=9;
    static int M[][]=new int[tam][tam];
    public static void main(String arg[]){

        M[3][2]=1;
        M[3][6]=1;
        M[2][0]=1;
        M[2][1]=1;
        M[1][4]=1;
        M[6][5]=1;
        M[6][7]=1;
        M[7][8]=1;

        prof(3);
    }
    static void prof(int raiz){
        Queue<Integer>c=new LinkedList<Integer>();

        c.add(raiz);
        while(!c.isEmpty()){
            int x=c.remove();
            int con=0;
            for(int i=0;i<tam;i++){
                if(M[x][i]==1){
                    c.add(i);
                    con++;
                }
            }
            if(con==0)
                System.out.print(x+" ");
        }
    }
}
```

Programa que muestra las hojas de un árbol del nivel 2, con recorrido en anchura
Se hace el uso de una segunda cola que almacena el nivel de cada nodo

```
import java.util.Queue;
import java.util.LinkedList;
public class Ejemplo1 {

    static int tam=9;
    static int M[][]=new int[tam][tam];
    public static void main(String arg[]){

        M[3][2]=1;
        M[3][6]=1;
        M[2][0]=1;
        M[2][1]=1;
        M[1][4]=1;
        M[6][5]=1;
        M[6][7]=1;
        M[7][8]=1;

        prof(3);
    }
    static void prof(int raiz){
        Queue<Integer>c=new LinkedList<Integer>();
        Queue<Integer>nivel=new LinkedList<Integer>();
        c.add(raiz);
        nivel.add(0);
        while(!c.isEmpty()){
            int x=c.remove();
            int n=nivel.remove();
            int con=0;
            for(int i=0;i<tam;i++){
                if(M[x][i]==1){
                    c.add(i);
                    nivel.add(n+1);
                    con++;
                }
            }
            if(con==0 & n==2)
                System.out.print(x+" ");
        }
    }
}
```

Programa que muestra todas las ramas de un árbol
Para este programa es necesario el recorrido en profundidad
Se hace uso de una segunda pila que almacena el nivel de cada nodo
Se hace uso de un vector que almacena los datos de la rama

```
import java.util.Stack;
public class Ejemplo1 {

    static int tam=9;
    static int M[][]=new int[tam][tam];
    public static void main(String arg[]){

        M[3][2]=1;
        M[3][6]=1;
        M[2][0]=1;
        M[2][1]=1;
        M[1][4]=1;
        M[6][5]=1;
        M[6][7]=1;
        M[7][8]=1;

        ramas(3);
    }
    static void ramas(int raiz){
        Stack<Integer>p=new Stack<Integer>();
        Stack<Integer>nivel=new Stack<Integer>();

        int vector[]=new int[tam];

        p.push(raiz);
        nivel.push(0);
        while(!p.isEmpty()){
            int x=p.pop();
            int n=nivel.pop();
            vector[n]=x;
            int con=0;
            for(int i=tam-1;i>=0;i--){
                if(M[x][i]==1){
                    p.push(i);
                    nivel.push(n+1);
                    con++;
                }
            }
            if(con==0){
                for(int i=0;i<=n;i++)
                    System.out.print(vector[i]+" ");
                System.out.println();
            }

        }
    }
}
```