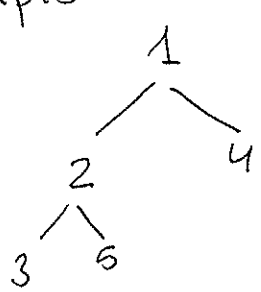
LECCION 20 : Arboles Binarios

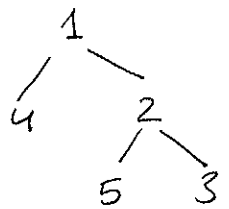
Ejercicio 1 : Obtener el recorrido en preorden con un preorden-iterator de un arbol Binario de enteros.

```
void RecorridoPreorden ( ArbolBinario<int> &ab){
    ArbolBinario<int>::preorden_iterator put;
    for (put = ab.beginpreorden(); put != ab.endpreorden(); ++put)
            cout << (*put) << " ";

}
void RecorridoInorden (ArbolBinario<int> &ab) {
    ArbolBinario<int>::inorden_iterator in_it;
    for (in_it=ab.begininorden(); in_it != ab.endinorden(); ++in_it)
            cout << (*in_it) << " ";

}
// igual para postorden.
```

Ejercicio 2 .- Dados dos arboles binarios de enteros implementar una funcion para establecer si dos árboles son reflejados o no

ejemplo





son reflejados.

```
bool Reflejados ( ArbolBinario<int>::nodo n1, ArbolBinario<int>::nodo n2
{
    if (n1.nulo() && n2.nulo())
        return true;
    else {
        if (*n1) != (*n2) ) return false
        else {
            return Reflejados (n1.hi(), n2.hd()) &&
                   Reflejados (n1.hd(), n2.hi());
}
```

(2) LECCION 20 : Arboles Binarios.

Ejercicio 3 : Obtener el arbol reflejado de uno dado

```
template <class T>
void RefljaArbol (ArbolBinario <T> &ab, typename ArbolBinario <T>::nodo
                                                              pos) {

    if (!pos.nulo() )
      if (! (pos.hi().nulo() && pos.hd().nulo() ) {
          ArbolBinario <T> ramai = ab.PodarHi_GetSubtree (pos);
          ArbolBinario <T> ramad = ab.PodarHd_GetSubtree (pos);
          ab.Insertar_Hd (pos, ramai) ;
          ab.Insertar_Hi (pos, ramad);
          typename ArbolBinario <T>::nodo pi = pos.hi();
          typename ArbolBinario <T>::nodo pd = pos.hd();
          RefljaArbol( ab, pi);
          RefljaArbol ( ab, pd);
      }

}
```

Version con el preorden y postorden para saber si dos arboles son reflejados.

```
bool SonReflejados (ArbolBinario <char> &a1, ArbolBinario <char> &a2){
    ArbolBinario <char> :: preorden_iterator pre_it;
    ArbolBinario <char> :: postorden_iterator post_it;
    string s1 = "", s2 = "";
    for ( pre_it = a1.beginpreorden(), post_it = a2.beginpostorden();
          pre_it != a1.endpreorden(), post_it != a2.endpostorden();
          ++pre_it, ++post_it){
                s1 += *pre_it;
                s2 += *post_it;
    }
    if (s1.size() != s2.size()) return false
    else{
        for (int i=0; i<s1.size() ; i++)
            if (s1[i] != s2[s2.size()-1-i]) return false;

        return true;
    }
}
```

3.