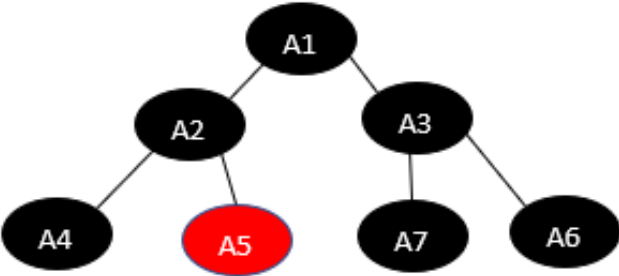


TAD Arbol Rojo Negro	
ARN = {A1,A2,A3...AN}	
A1 es el principal elemento, A2 y A3 son subarboles de A1, cualquier elemento menor a A1 va hacia la izquierda, y si es mayor a A1 va a la derecha.	
El arbol RN tiene un comportamiento auto balanceable que tiene que ver con su bit extra que significa color.	
 <p>$A_4 < A_2 < A_5 < A_1 < A_6 < A_3 < A_7$</p>	
{inv: (1) Cada nodo tiene un color rojo o negro. (2) La raíz del árbol es siempre negra. (3) No hay dos nodos rojos adyacentes (un nodo rojo no puede tener un padre rojo o un hijo rojo) (4) Cada ruta desde un nodo (incluida la raíz) a cualquiera de sus nodos NULL descendientes tiene el mismo número de nodos negros.}	
createRBT:	-> RBT
insert:	Element x RBT -> RBT
remove:	Element x RBT -> RBT
search:	RBT -> Element
rotateLeft:	Element x RBT -> RBT
rotateRight:	Element x RBT -> RBT
rotateLeftRepair:	Element x RBT -> RBT
rotateRightRepair:	Element x RBT -> RBT
size:	Element x RBT -> RBT

insert(K key, V value):Modifier
"Insertar una nueva llave dentro del arbol, si la llave ya existe, se inserta una nueva posicion"
{ pre: Arbol RN inicializado }
{ post: Nuevo valor insertado al arobl }
remove(K key):Modifier
"Eliminar un valor o clave especifico del Arbol RN"
{ pre: Arbol RN inicializado }
{ post: Valor o llave especifica eliminado del arbol }
search(K key): Analyzer
"Busca un valor de clave especifico dentro del arbol y lo devuelve"
{ pre: Arbol RN inicializado }
{ post: Devuelve un ArrayList de valores }
rotateLeft(Element): Modifier
"Se hace la rotacion del arbol hacia la izquierda para balancear el arbol RN"
{ pre: Arbol RN inicializado y los objetivos de rotacion deben de existir != null }
{ post: El arbol de estructura binaria modificada con la rotacion }
rotateRight(Element): Modifier
"Se hace la rotacion del arbol hacia la derecha para balancear el arbol RN"
{ pre: Arbol RN inicializado y los objetivos de rotacion deben de existir != null }
{ post: El arbol de estructura binaria modificada con la rotacion }

rotateLeftRepair(Element): Modifier
"Se hace la rotacion del arbol hacia la izquierda cuando el arbol RN no esta balanceado"
{ pre: Arbol RN inicializado y los objetivos de rotacion deben de existir != null }
{ post: El arbol de estructura binaria modificada con la rotacion }
rotaterRightRepair(Element): Modifier
"Se hace la rotacion del arbol hacia la derecha cuando el arbol RN no esta balanceado"
{ pre: Arbol RN inicializado y los objetivos de rotacion deben de existir != null }
{ post: El arbol de estructura binaria modificada con la rotacion }
size(Element): Analyzer
"Size"
{ pre: Arbol RN inicializado y los objetivos de rotacion deben de existir != null }
{ post: Entero que representa el size del ARN }