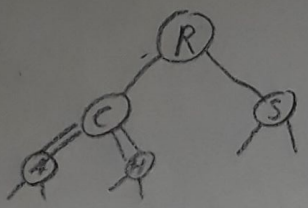
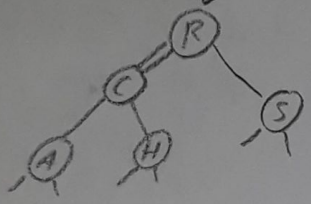
```
private:
   int red(link x){
      if(x==0)                  // si se envia nulo
        return 0;               // se regresa 0.
      return x->red;            // si no es nulo, se regresa el valor indicado.
   }
   void RBinsert(link &h, Item x, int sw){
(1)  if(h==0){                  // si se envia nulo
(2)     h=new node(x);          // crea un nuevo nodo con valor x
(3)     return;                 // sal de la función
        }
(4)  if(red(h->l) && red(h->r)){ // si los dos hijos son rojos
(5)     h->red=1;               // se hace rojo este nodo
(6)     h->l->red=0;            // el hijo izquierdo se hace negro
(7)     h->r->red=0;            // el hijo derecho se hace negro
        }
(8)  if(x.key()<h->item.key()){ // si lo enviado es menor al valor del nodo actual
(9)     RBinsert(h->l, x, 0);   // llama la función con el subarbol izquierdo
(10)    if(red(h) && red(h->l) && sw) // si todos son rojos.
(11)       rot+R(h);            // rota a la derecha h.
(12)    if(red(h->l) && red(h->l->l)){ // si los dos nodos izquierdos son rojos
(13)       rot R(h);            // rota a la derecha
(14)       h->red=0             // hace el nodo actual negro
(15)       h->r->red=1;         // hace el nodo de la derecha rojo
        }
      }
(16)  else{
(17)    RBinsert(h->r, x, 1);   // llama a la función con el derecho rojo.
(18)    if(red(h) && red(h->r) && !sw) // si el nodo actual y el derecho son rojos.
(19)       rot+L(h);            // rota a la izquierda.
(20)    if(red(h->r) && red(h->r->r)){ // si los dos nodos derechos son rojos
(21)       rot+L(h);            // rota a la izquierda
(22)       h->red=0;            // el nodo actual se hace negro
(23)       h->l->red=1;         // el hijo izquierdo es rojo.
        }
      }
   }
```

```
Public:
    void insert(Item x){
(24)    RBinsert(head, x, 0); // llama la funcion
(25)    head->red=0;          // la cabeza es negra.
    }
```

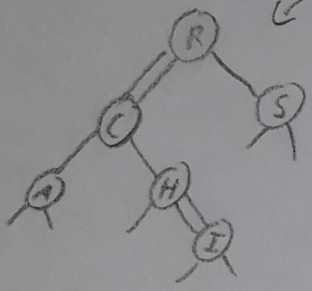
Secuencia:



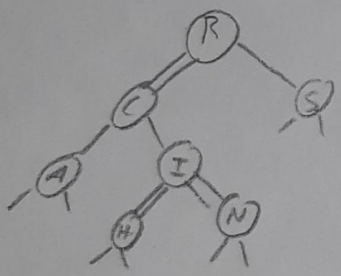
(24), (8), (9), (4), (5), (6), (7),



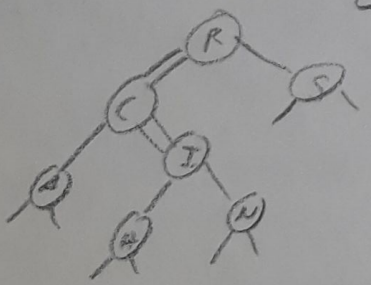
(16), (17), (16), (17), (1), (2), (3)



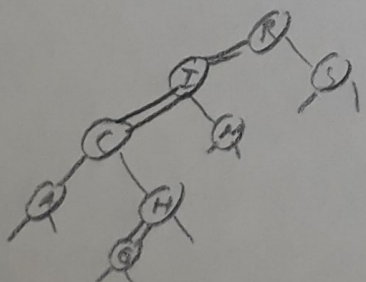
(23)



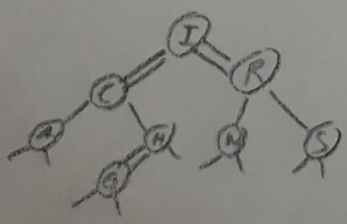
(24), (8), (9), (8), (9), (4), (5), (6), (7),



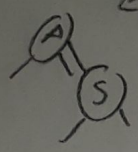
(8), (9), (8), (9), (1), (2), (3), (10), (11),



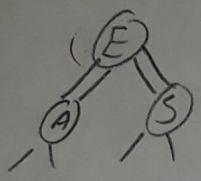
(10), (11).



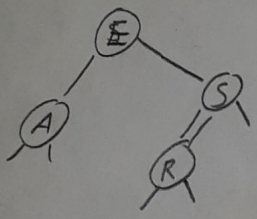
(A) (24), (16), (17), (1), (2), (3)



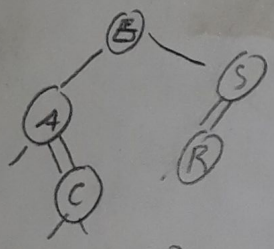
(24), (16), (17), (8), (9), (1), (2), (3), (10), (11), (18), (19)



(24), (4), (5), (6), (7), (16), (17), (8), (9), (1), (2), (3),



(24), (8), (9), (16), (17), (1), (2), (3)



(24), (16), (17), (8), (9), (8), (9), (1), (2), (3), (10), (11)