Algorithm Design
M. T. Goodrich and R. Tamassia
John Wiley & Sons
Solution of Exercise C-3.17

We give two possible solutions. Let v be a red node of a red-black tree T.

- This solution assumes that keys in T are integers. We change keys stored in T using the following simple rule: if v is red, set $\mathsf{new\_key}(v) = 2\mathsf{key}(v)+1$, or if v is black, set $\mathsf{new\_key}(v) = 2\mathsf{key}(v)$. Observe that this "mapping" function is 1-1 and strictly increasing; this property preserves the property of T of being a binary search tree and helps in uniquely identify a node's color. Namely, any node can be identified of being red or black, by simply checking whether $\mathsf{new\_key}(v)$ is respectively odd or even.
- This solution assumes that keys in T are unique. Because of the fact that v is red, it can not be a leaf. Let u and w be v's left and right child respectively. If both u and w are not place-holder nodes, we can represent the fact that v is red, implicitly, by exchanging the children of v, i.e., by making u v's right child and w v's left child. This allows us to correctly identify if a node is red or black, only by checking the key values for nodes u, v and w. Node v is red, if and only if the property of T of being binary search is locally violated, i.e., if and only if $\mathsf{key}(u) > \mathsf{key}(v) > \mathsf{key}(w)$. If one of v's children is a place holder node, then again a similar violation of T's property of being a binary tree can be used to identify v's color. Finally, if both children of v are place-holder nodes, then we can either store some fictitious element in one of them or remove v's right (or left) child.