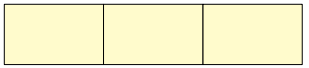
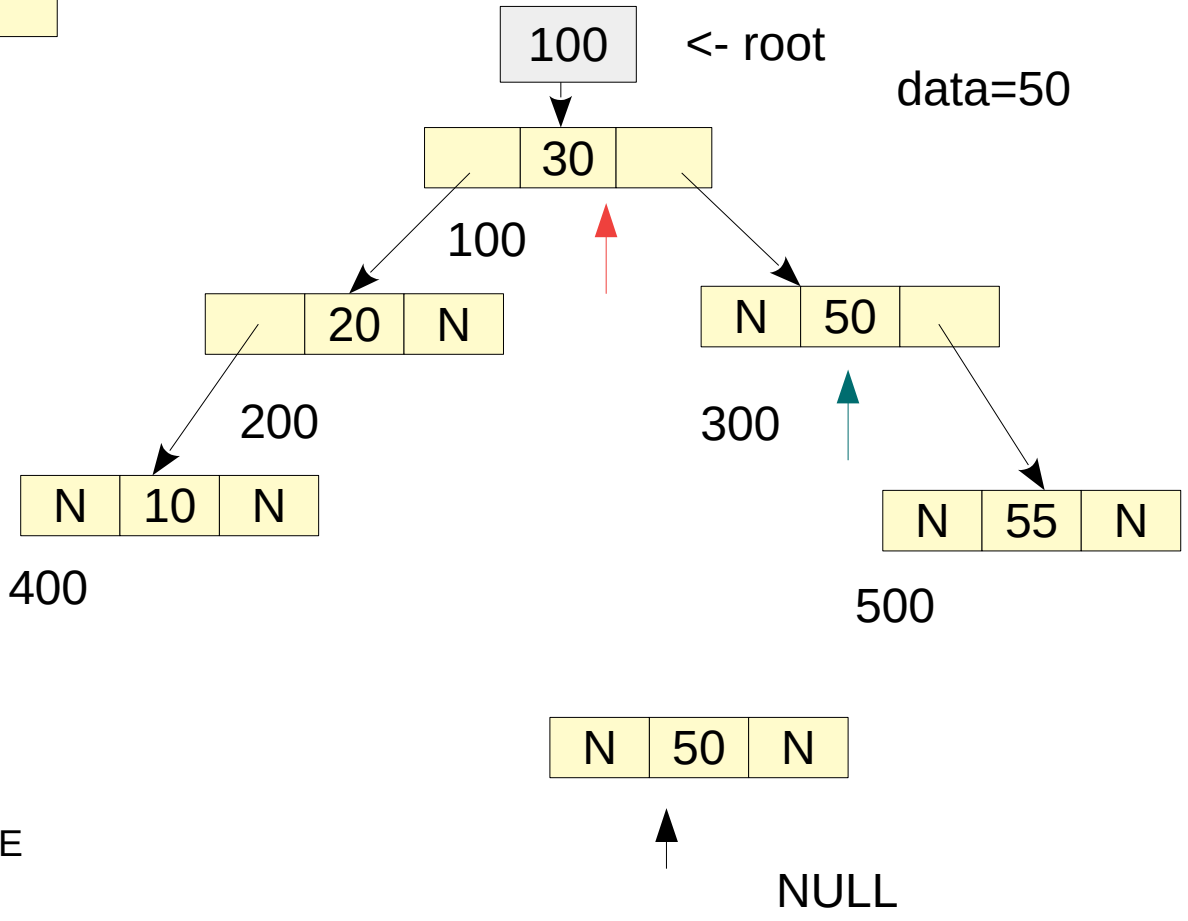
```
create_node(data)
{
	new = Memalloc(sizeof(tree_t))
	if (new = NULL)
		return FAILURE / e_false
	new -> data = data
	new -> LC = new -> RC = NULL
	return new
}
```

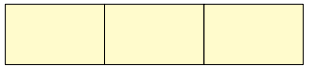
BST_insert(root, data)

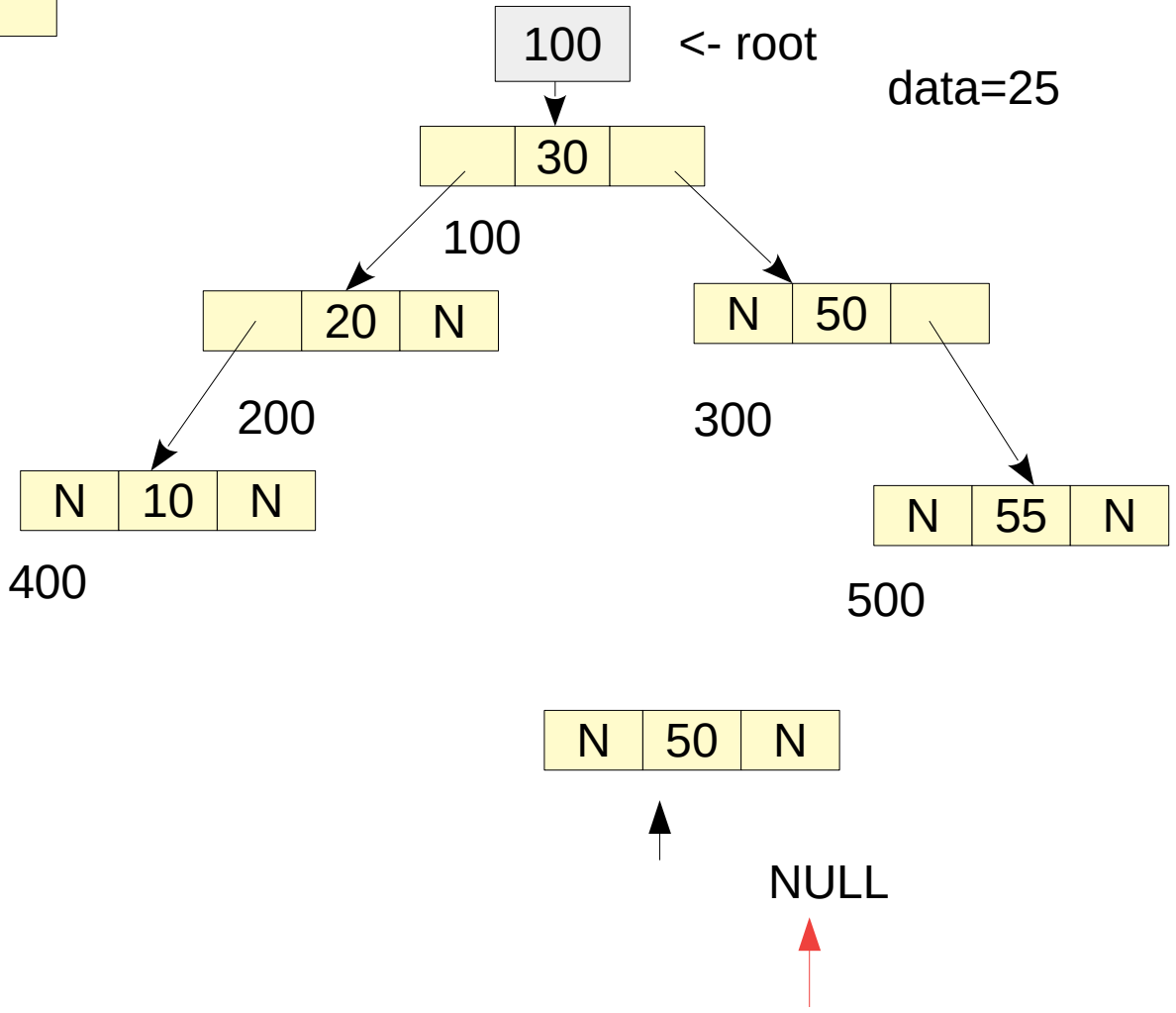


```
1. new = Memalloc(sizeof(tree_t))
2. if (new = NULL)
                return FAILURE / e_false
3. new -> data = data
4. new -> LC = new -> RC = NULL
5. if (root = NULL)
        root = new;
        return SUCCESS / e_true
temp = root
6. while (temp != NULL)
{
        if (data < temp -> data)
                parent = temp
                temp = temp -> LC
        else if (data > temp -> data)
                parent = temp
                temp = temp -> RC
        else
                free(new),  return DUPLICATE / FAILURE
}
7. if(parent -> data > data)
        parent -> LC = new
    else
        parent -> RC = new
8. return SUCCESS / e_true
```



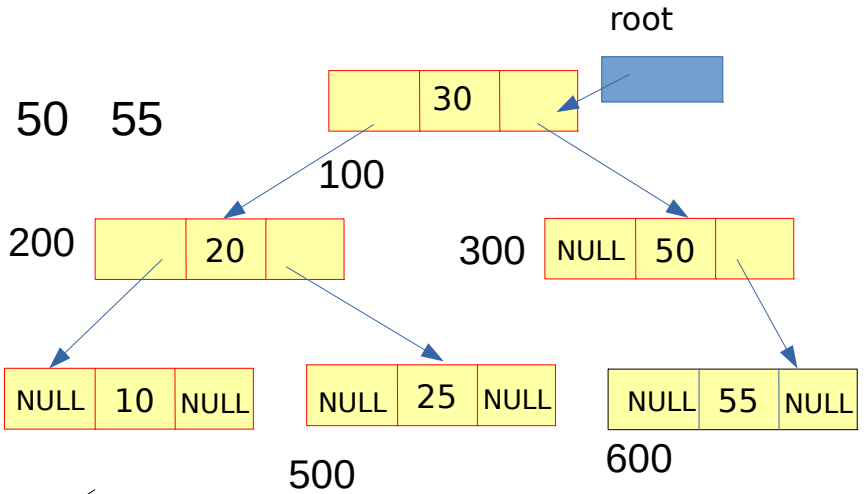
```
BST_search(root, data)

1. if (root = NULL)
        return FAILURE / TREE_EMPTY

temp = root
2. while (temp != NULL)
{
        if (data < temp -> data)
                temp = temp -> LC
        else if (data > temp -> data)
                temp = temp -> RC
        else
                return DATA_FOUND
}
3. return DATA_NOT_FOUND
```

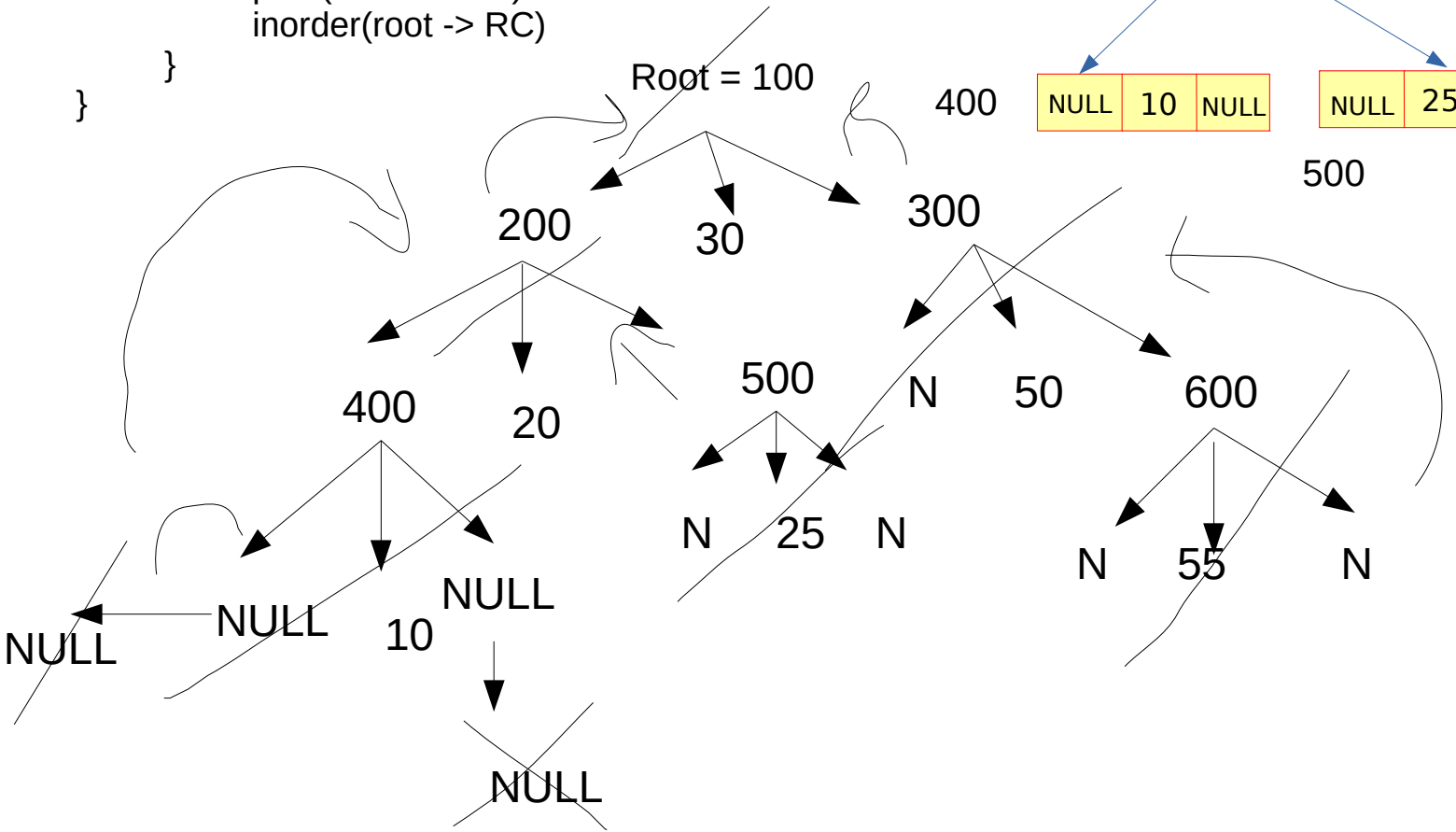




```
inorder(root)
{
        if (root != NULL)
        {
                inorder(root -> LC)
                print(root -> data)
                inorder(root -> RC)
        }
}
```

10 20 25 30 50 55





```
findmin(root)
{
        while (root -> LC)
                root = root -> LC

        return root -> data
}
```