Software Verification
Exercise Solution: Separation Logic

By application of small axioms and the frame rule, we obtain the following proof outline:

```
copytree(i; j) =
    {tree τ i}
    if i = nil then
        {tree τ i ∧ i = nil}
        j := i
        {tree τ i ∧ i = nil ∧ j = i}    // ... by e.g. Hoare's forward assignment axiom.
        {empty ∧ τ = ε ∧ i = nil ∧ j = nil * empty}
        // Facts that do not involve the heap can migrate over * or be duplicated over it:
        {empty ∧ τ = ε ∧ i = nil * empty ∧ τ = ε ∧ j = nil}
        {tree τ i * tree τ j}
    else
        newvar i1, i2, v, j1, j2 in
            {tree τ i ∧ i != nil}
            {∃j,a,k,τ1,τ2. (i↦j,a,k) * (tree τ1 j) * (tree τ2 k) ∧ τ = (τ1,a,τ2)}
            i1 := [i];
            {∃a,k,τ1,τ2. (i↦i1,a,k) * (tree τ1 i1) * (tree τ2 k) ∧ τ = (τ1,a,τ2)}
            v := [i + 1];
            {∃k,τ1,τ2. (i↦i1,v,k) * (tree τ1 i1) * (tree τ2 k) ∧ τ = (τ1,v,τ2)}
            i2 := [i + 2];
            {∃τ1,τ2. (i↦i1,v,i2) * (tree τ1 i1) * (tree τ2 i2) ∧ τ = (τ1,v,τ2)}
            copytree(i1, j1);
            {∃τ1,τ2. (i↦i1,v,i2) * (tree τ1 i1) * (tree τ1 j1) * (tree τ2 i2) ∧ τ = (τ1,v,τ2)}
            copytree(i2, j2);
            {∃τ1,τ2. (i↦i1,v,i2) * (tree τ1 i1) * (tree τ1 j1) * (tree τ2 i2) * (tree τ2 j2) ∧ τ =
(τ1,v,τ2)}
            j := cons(j1, v, j2);
            {∃τ1,τ2. (i↦i1,v,i2) * (tree τ1 i1) * (tree τ1 j1) * (tree τ2 i2) * (tree τ2 j2) *
(j↦j1,v,j2) ∧ τ = (τ1,v,τ2)}
            {∃τ1,τ2. (i↦i1,v,i2) * (tree τ1 i1) * (tree τ2 i2) * (j↦j1,v,j2) * (tree τ1 j1) *
(tree τ2 j2) ∧ τ = (τ1,v,τ2)}
            {tree τ i * tree τ j}
        end
    end
    {tree τ i * tree τ j}
```

Remarks:

There are, as usual, several proofs for the correctness of a single code snippet. For example, we can prove the first branch of the if-statement as follows:

$\{\text{tree } \tau \text{ i} \wedge \text{i} = \text{nil}\}$
$\{\tau = \varepsilon \wedge \text{empty} \wedge \text{i} = \text{nil}\}$
$\{\tau = \varepsilon \wedge (\text{empty} \wedge \text{i} = \text{nil}) * (\text{empty} \wedge \text{i} = \text{nil})\}$ // By e.g. Hoare's backward axiom:
j := i
$\{\tau = \varepsilon \wedge (\text{empty} \wedge \text{i} = \text{nil}) * (\text{empty} \wedge \text{j} = \text{nil})\}$
$\{\tau = \varepsilon \wedge (\text{tree } \varepsilon \text{ i}) * (\text{tree } \varepsilon \text{ j})\}$
$\{\text{tree } \tau \text{ i} * \text{tree } \tau \text{ j}\}$

The proof of this code snippet uses only familiar rules of Hoare logic: assignment and consequence. The implications used by the rule of consequence are of course now expressed in separation logic.

The next part of the proof employs the small axioms and the frame rule. It is convenient to use the following derived axiom for heap lookup:

$\{e \mapsto e'\}$ x := [e] $\{e \mapsto e' \wedge x = e'\}$
provided x does not appear free in e or e'.

Here is a detailed proof of the first heap lookup:

$\{\text{tree } \tau \text{ i} \wedge \text{i} \mathrel{!=} \text{nil}\}$
$\{\exists j, a, k, \tau_1, \tau_2.\ (i \mapsto j, a, k) * (\text{tree } \tau_1\ j) * (\text{tree } \tau_2\ k) \wedge \tau = (\tau_1, a, \tau_2)\}$
$\{\exists a, k, \tau_1, \tau_2.\ (i \mapsto j, a, k) * (\text{tree } \tau_1\ j) * (\text{tree } \tau_2\ k) \wedge \tau = (\tau_1, a, \tau_2)\}$
$\{i \mapsto j * \exists a, k, \tau_1, \tau_2.\ (i+1 \mapsto a, k) * (\text{tree } \tau_1\ j) * (\text{tree } \tau_2\ k) \wedge \tau = (\tau_1, a, \tau_2)\}$
$\{i \mapsto j\}$
i_1 := [i]
$\{i \mapsto j \wedge i_1 = j\}$
$\{(i \mapsto j \wedge i_1 = j) * \exists a, k, \tau_1, \tau_2.\ (i+1 \mapsto a, k) * (\text{tree } \tau_1\ j) * (\text{tree } \tau_2\ k) \wedge \tau = (\tau_1, a, \tau_2)\}$
$\{\exists a, k, \tau_1, \tau_2.\ (i \mapsto j, a, k \wedge i_1 = j) * (\text{tree } \tau_1\ j) * (\text{tree } \tau_2\ k) \wedge \tau = (\tau_1, a, \tau_2)\}$
$\{\exists j, a, k, \tau_1, \tau_2.\ (i \mapsto j, a, k \wedge i_1 = j) * (\text{tree } \tau_1\ j) * (\text{tree } \tau_2\ k) \wedge \tau = (\tau_1, a, \tau_2)\}$
$\{\exists j, a, k, \tau_1, \tau_2.\ (i \mapsto i_1, a, k) * (\text{tree } \tau_1\ i_1) * (\text{tree } \tau_2\ k) \wedge \tau = (\tau_1, a, \tau_2)\}$
$\{\exists a, k, \tau_1, \tau_2.\ (i \mapsto i_1, a, k) * (\text{tree } \tau_1\ i_1) * (\text{tree } \tau_2\ k) \wedge \tau = (\tau_1, a, \tau_2)\}$

Note that we applied Auxiliary Variable Elimination to quantify only j - the other variables were quantified in the frame. In contrast to this, the following detailed proof of the first recursive call to copytree uses AuxVarElim to quantify both τ_1 and τ_2. It includes no quantifiers in the frame:

$\{\exists\tau_1,\tau_2.\ (i \mapsto i_1,v,i_2) * (\text{tree } \tau_1\ i_1) * (\text{tree } \tau_2\ i_2) \wedge \tau = (\tau_1,v,\tau_2)\}$

$\{\text{tree } \tau_1\ i_1 * (i \mapsto i_1,v,i_2) * (\text{tree } \tau_2\ i_2) \wedge \tau = (\tau_1,v,\tau_2)\}$

$\{\text{tree } \tau_1\ i_1\}$

copytree(i_1, j_1)

$\{\text{tree } \tau_1\ i_1 * \text{tree } \tau_1\ j_1\}$

$\{\text{tree } \tau_1\ i_1 * \text{tree } \tau_1\ j_1 * (i \mapsto i_1,v,i_2) * (\text{tree } \tau_2\ i_2) \wedge \tau = (\tau_1,v,\tau_2)\}$

$\{(i \mapsto i_1,v,i_2) * (\text{tree } \tau_1\ i_1) * (\text{tree } \tau_1\ j_1) * (\text{tree } \tau_2\ i_2) \wedge \tau = (\tau_1,v,\tau_2)\}$

$\{\exists\tau_1,\tau_2.\ (i \mapsto i_1,v,i_2) * (\text{tree } \tau_1\ i_1) * (\text{tree } \tau_1\ j_1) * (\text{tree } \tau_2\ i_2) \wedge \tau = (\tau_1,v,\tau_2)\}$

Here is a detailed proof of the final cons command. It is similar to the previous proof because it excludes existential quantifies from the frame:

$\{\exists\tau_1,\tau_2.\ (i \mapsto i_1,v,i_2) * (\text{tree } \tau_1\ i_1) * (\text{tree } \tau_1\ j_1) * (\text{tree } \tau_2\ i_2) * (\text{tree } \tau_2\ j_2) \wedge \tau = (\tau_1,v,\tau_2)\}$

$\{(i \mapsto i_1,v,i_2) * (\text{tree } \tau_1\ i_1) * (\text{tree } \tau_1\ j_1) * (\text{tree } \tau_2\ i_2) * (\text{tree } \tau_2\ j_2) \wedge \tau = (\tau_1,v,\tau_2)\}$

$\{\text{empty} * (i \mapsto i_1,v,i_2) * (\text{tree } \tau_1\ i_1) * (\text{tree } \tau_1\ j_1) * (\text{tree } \tau_2\ i_2) * (\text{tree } \tau_2\ j_2) \wedge \tau = (\tau_1,v,\tau_2)\}$

$\{\text{empty}\}$

$j := \text{cons}(j_1, v, j_2)$

$\{j \mapsto j_1,v,j_2\}$

$\{j \mapsto j_1,v,j_2 * (i \mapsto i_1,v,i_2) * (\text{tree } \tau_1\ i_1) * (\text{tree } \tau_1\ j_1) * (\text{tree } \tau_2\ i_2) * (\text{tree } \tau_2\ j_2) \wedge \tau = (\tau_1,v,\tau_2)\}$

$\{i \mapsto i_1,v,i_2 * (\text{tree } \tau_1\ i_1) * (\text{tree } \tau_1\ j_1) * (\text{tree } \tau_2\ i_2) * (\text{tree } \tau_2\ j_2) * (j \mapsto j_1,v,j_2) \wedge \tau = (\tau_1,v,\tau_2)\}$

$\{\exists\tau_1,\tau_2.\ (i \mapsto i_1,v,i_2) * (\text{tree } \tau_1\ i_1) * (\text{tree } \tau_1\ j_1) * (\text{tree } \tau_2\ i_2) * (\text{tree } \tau_2\ j_2) * (j \mapsto j_1,v,j_2) \wedge \tau = (\tau_1,v,\tau_2)\}$