Software Verification
Exercise Solution: Hoare & Separation logic

Solutions:

1) We introduce the helper variable z. The loop invariant y = z! is crucial for the proof, and can be found by executing the loop for a couple of iterations with test values.

{true}
{1 = 0!}
y := 1;
{y = 0!}
z := 0;
{y = z!}
while (z != x) {
    {y = z! ∧ z ≠ x}
    {y.(z + 1) = (z + 1)!}
    z := z + 1;
    {y.z = z!}
    y := y * z;
    {y = z!}
}
{y = z! ∧ ¬(z ≠ x)}
{y = x!}

2) The algorithm employs a helper variable k. The proof uses the definition of the list predicate from the slides (see slide 18). We first give a rather detailed proof outline. You can use fewer assertions in the exam - the second outline is a good example.

{list (a::as) i}
{∃j. i↦a,j * list as j}
    {i↦a,j * list as j}
        {i↦a}
        dispose(i);
        {empty}
    {i+1↦j * list as j}
        {i+1↦j}
        k := [i+1];
        {i+1↦j ∧ k=j}
    {i+1↦j * list as j ∧ k=j}
        {i+1↦j}
        dispose(i+1);
        {empty}
    {list as j ∧ k=j}
{∃j. list as j ∧ k=j}
{list as k}
        i := k
{list as i}

{list (a::as) i}
{∃j. i↦a,j * list as j}
    {i↦a,j * list as j}
    dispose(i);
    {i+1↦j * list as j}
    k := [i+1];
    {i+1↦j * list as j ∧ k=j}
    dispose(i+1);
    {list as j ∧ k=j}
{∃j. list as j ∧ k=j}
{list as k}
    i := k
{list as i}