Mid-term Exam
Sample Solutions

Problem 1.
In this code the upper and lower bounds for the binary search range are variables 'high' and 'low', respectively. On each iteration, the item at the midpoint (mid) is tested. If it is the target value x, the search ends, otherwise either the upper or lower bound is shifted to the midpoint and another iteration is performed. When the range has collapsed to a single item, the iteration is terminated and then the final test for x is made.

```
{n≥1 ⇒ ∀i:N • 1≤i<n ⇒ A[i]≤A[i+1]}
low:= 1;  high:= n;  mid:= (n+1) div 2;
  {INV: 1≤low≤mid≤high≤n ∧ (∀i:N • 1≤i<n ⇒ A[i]≤A[i+1])
        ∧ ∀i:N • (1≤i<low   high<i≤n) ⇒ A[i]≠x}
  while (low<high) ∧ (x≠A[mid]) do
  begin
  if x>A[mid]   low=mid
    then low:= mid+1
    else high:= mid-1;
  mid:= (low+high) div 2;
  end;
if A[mid]=x then k:= mid else k:= 0;
{(1≤k≤n ∧ A[k]=x)   (k=0 ∧ ∀i:N • 1≤i≤n ⇒ A[i]≠x)}
```

The proof rule for loops allows us to conclude the loop invariant together with the negation of the loop guard after a loop. Therefore the conjunction of these two conditions must be sufficient to justify the desired conclusions. Also, the loop body requires that A is sorted, and the loop invariant is the main part of the pre/post-conditions of the loop, so to prove using any assertion as the invariant, “A sorted” must be a part of that assertion.

This code makes some concessions to proving. To cover all cases of the index, we use the negation of the loop guard to infer low≥high, and to force equality at the end we need the converse in the loop invariant. One subtle place in this code occurs when high = low+1. In this case, mid is equal to low, and if high is decremented to mid-1, the code still works, but the invariant doesn’t since then high<low. In the code above, this situation is detected in the if-statement within the loop by the low=mid test, and the troublesome step is avoided.

The loop invariant asserts that variables low, mid and high are ordered, A is sorted, and for indices outside the range low..high, item x is not present in the array. This reflects the basic strategy of binary search — the range of indices known to *exclude* x is initially empty, and half of the remaining range is added at each iteration. Therefore when the loop terminates, using the invariant and the negated loop guard, we can infer that:

- either A[mid]=x (negated loop guard) and 1≤low≤mid≤high≤n (invariant), and k is subsequently set accordingly,
- or x≠A[mid] so low=mid=high ∧ ∀i:N • (1≤i<low   high<i≤n) ⇒ A[i]≠x (negated loop guard and invariant) so x is not present in A, and k will be set to 0.

Problem 2.
The solution to this problem is expressed using ZANS in a file in our class directory.

Problem 3.
(a) For relational image, $f(\!(A_1 \cup A_2)\!) = \{x: \text{dom } f; y: \text{ran } f \mid (x,y) \in f \bullet x \in A_1 \cup A_2 \bullet y\}$. But this means $x \in A_1$ and $x \in A_2$ so that $y \in f(\!(A_1)\!)$ and $y \in f(\!(A_2)\!)$. Therefore $f(\!(A_1 \cup A_2)\!) \subseteq f(\!(A_1)\!) \cap f(\!(A_2)\!)$. Note that it is immaterial that f is an injection for these steps

Conversely, since $y \in f(\!(A_1)\!) \cap f(\!(A_2)\!)$, $y \in f(\!(A_1)\!)$ and $y \in f(\!(A_2)\!)$. And if $y \in f(\!(A_1)\!)$, there exists $x_1 \in A_1$, and $(x_1,y) \in f$. Similarly, there exists $x_2 \in A_2$, and $(x_2,y) \in f$. But since f is an injection, this implies that $x_1 = x_2$. Let $x_1 = x_2 = x$, and this means that $x \in A_1 \cap A_2$, and hence $y \in f(\!(A_1 \cap A_2)\!)$. Therefore $f(\!(A_1)\!) \cap f(\!(A_2)\!) \subseteq f(\!(A_1 \cap A_2)\!)$, and the proof is complete.

(b) The identity fails for general partial functions. For $f: \{1,2\} \to \{1,2\}$ defined by $f = \{(1,1), (2,1)\}$, let $A_1 = \{1\}$ and $A_2 = \{2\}$. Then $f(\!(A_1 \cap A_2)\!) = f(\!(\varnothing)\!) = \varnothing$, while $f(\!(A_1)\!) \cap f(\!(A_2)\!) = \{1\} \cap \{1\} = \{1\}$. However, $f(\!(A_1 \cap A_2)\!) \subseteq f(\!(A_1)\!) \cap f(\!(A_2)\!)$ remains true as noted in (a) above.

Problem 4.
The solution to this problem is expressed using ZANS in a file in our class directory.