Forritunarmál Einstaklingsverkefni 2

brj46

September 2024

https://tinyurl.com/benjaminreynir

Dafny

```
// For k>=0 this function returns 1+2+3+...+k.
// This is the sum of the first k integers >0.
// If k==0 then this sum is 0.
// In older versions of Dafny a function like this
// is not executable but can take part in program
// verification and the Dafny compiler "understands"
// the body of the function.
function SumInts(k: int): int
  requires k >= 0
{
  if k == 0 then 0 else SumInts(k-1) + k
}

// Compute SumInts using a loop and prove
// that SumInts(k) == (k+1)*k/2.
method SumIntsLoop(k: int) returns (s: int)
  requires k >= 0
  ensures s == (k+1)*k/2
  ensures s == SumInts(k)
{
  var i := 0;
  s := 0;
  while i < k
    invariant 0 <= i <= k
    decreases k - i
    invariant s == SumInts(i)
    invariant s == i*(i+1)/2
  {
    i := i + 1;
    s := s + i;
  }
}
```

```
// Compute SumInts using recursion and prove
// that SumInts(k) == (k+1)*k/2.
method SumIntsRecursive(k: int) returns (s: int)
  requires k >= 0
  ensures s == (k+1)*k/2
  ensures s == SumInts(k)
{
  if k == 0 {
    s := 0;  // Base case: sum of first 0 numbers is 0
  } else {
    var a := SumIntsRecursive(k-1);  // Recursively get the sum of k-1
    s := a + k;  // Add k to the sum
  }
}

// Compute SumInts using tail recursion and prove
// that SumInts(k) == (k+1)*k/2.
method SumIntsTailRecursive(i: int, r: int, k: int) returns (s: int)
  requires 0 <= i <= k
  decreases k - i
  requires r == (i+1)*i/2
  requires r == SumInts(i)
  ensures s == (k+1)*k/2
  ensures s == SumInts(k)
{
  if i == k {
    s := r;  // If i equals k, return the accumulated sum
  } else {
    var tailSum := SumIntsTailRecursive(i+1, r+i+1, k);  // Accumulate sum and recurse
    s := tailSum;
  }
}

// Main method to test the functions.
method Main()
{
  var s1 := SumIntsLoop(5);
  var s2 := SumIntsRecursive(5);
  var s3 := SumIntsTailRecursive(0, 0, 5);
  var s4 := SumIntsTailRecursive(4, 10, 5);
  print s1;  // Expected: 15
  print s2;  // Expected: 15
  print s3;  // Expected: 15
  print s4;  // Expected: 15
}
```

1

```
function SumInts(k: int): int
  requires k >= 0
{
  if k == 0 then 0 else SumInts(k-1) + k
}

// Compute SumInts using a loop and prove
// that SumInts(k) == (k+1)*k/2.
method SumIntsLoop(k: int) returns (s: int)
  requires k >= 0
  ensures s == (k+1)*k/2
  ensures s == SumInts(k)
{
  var i := 0;
  s := 0;
  while i < k
    invariant 0 <= i <= k
    decreases k - i
    invariant s == SumInts(i)
    invariant s == i*(i+1)/2
  {
    i := i + 1;
    s := s + i;
  }
}
```

2

```
    // Compute SumInts using recursion and prove
// that SumInts(k) == (k+1)*k/2.
method SumIntsRecursive(k: int) returns (s: int)
  requires k >= 0
  ensures s == (k+1)*k/2
  ensures s == SumInts(k)
{
  if k == 0 {
    s := 0;  // Base case: sum of first 0 numbers is 0
  } else {
    var a := SumIntsRecursive(k-1);  // Recursively get the sum
        of k-1
    s := a + k;  // Add k to the sum
  }
}
```

3

```
    // Compute SumInts using tail recursion and prove
// that SumInts(k) == (k+1)*k/2.
method SumIntsTailRecursive(i: int, r: int, k: int) returns (s:
    int)
  requires 0 <= i <= k
  decreases k - i
  requires r == (i+1)*i/2
  requires r == SumInts(i)
  ensures s == (k+1)*k/2
  ensures s == SumInts(k)
{
  if i == k {
    s := r;  // If i equals k, return the accumulated sum
  } else {
    var tailSum := SumIntsTailRecursive(i+1, r+i+1, k);  //
       Accumulate sum and recurse
    s := tailSum;
  }
}
```