Alexander Guðmundsson
alg35@hi.is

```
// Höfundur spurningar:  Snorri Agnarsson, snorri@hi.is
// Permalink spurningar: https://rise4fun.com/Dafny/vkxp

// Höfundur lausnar:     Alexander Guðmundsson
// Permalink lausnar:    https://rise4fun.com/Dafny/HLplX

///////////////////////////////////////////////////////////////
// Hér byrjar óbreytanlegi hluti skrárinnar.
// Fyrir aftan þann hluta er sá hluti sem þið eigið að breyta.
///////////////////////////////////////////////////////////////

// IsSorted(a) er satt þá og því aðeins að
// sannað sé að a sé raðað í vaxandi röð.
predicate IsSorted( a: seq<int> )
{
   forall p,q | 0 <= p < q < |a| :: a[p] <= a[q]
}

// Sannar að poki með einu staki samsvarar runu
// með einu staki.  Dafny þarf smávegis olnbogaskot
// til að fatta það. Þetta er gagnlegt til að sanna
// að útkoman úr Sort sé rétt í sértilvikinu þegar
// raðað er poka m með aðeins einu gildi x, sem
// gefur þá rununa s == [x].
lemma Singleton( m: multiset<int>, s: seq<int>, x: int )
   requires x in m;
   requires x in s;
   requires |s| == 1 == |m|;
   ensures |m-multiset{x}| == 0;
   ensures s == [x];
   ensures m == multiset{x};
   ensures m == multiset(s);
   ensures IsSorted(s);
{}

method RemoveOne( a: multiset<int> ) returns( b: multiset<int>, x: int )
   requires |a| >= 1;
   ensures a == b+multiset{x};
{
   x :| x in a;
```

Alexander Guðmundsson
alg35@hi.is

```
   b := a-multiset{x};
}

// Þessi hjálparsetning er gagnleg til að hjálpa
// Dafny að sanna að útkoman úr röðuninni sé rétt.
lemma LomutoLemma    ( a: multiset<int>
                     , a': seq<int>
                     , x: int
                     , b: multiset<int>
                     , b': seq<int>
                     , c: seq<int>
                     )
   requires a == multiset(a');
   requires b == multiset(b');
   requires IsSorted(a');
   requires IsSorted(b');
   requires forall z | z in a :: z<=x;
   requires forall z | z in b :: z>=x;
   requires c == a'+[x]+b';
   ensures forall p | 0<=p<|a'| :: a'[p] in a;
   ensures forall p | 0<=p<|b'| :: b'[p] in b;
   ensures forall z | z in a' :: z in a && z<=x;
   ensures forall z | z in b' :: z in b && z>=x;
   ensures forall z | z in a' :: z in a && z<=x;
   ensures forall z | z in b' :: z in b && z>=x;
   ensures IsSorted(c);
   ensures multiset(c) == a+multiset{x}+b;
{
   assert |c| == |a'|+1+|b'|;
   assert forall p,q | 0<=p<q<|c| :: q<|a'| ==> c[p]<=c[q];
   assert forall p,q | 0<=p<q<|c| :: q==|a'| ==> c[q]==x && p<|a'| &&
c[p]==a'[p] && c[p] in a && c[p]<=c[q];
   assert forall p,q | 0<=p<q<|c| :: p<|a'| && q>|a'| ==> c[p] in a &&
c[q] in b && c[p]<=c[q];
   assert forall p,q | 0<=p<q<|c| :: p==|a'| && q>|a'| ==> c[p]==x && c[q]
in b && c[p]<=c[q];
   assert forall p,q | 0<=p<q<|c| :: p>|a'| && q>|a'| ==> c[p]<=c[q];
}

// Prófunarfall sem staðfestir að Partition og Sort
```

Alexander Guðmundsson
alg35@hi.is

```
// séu áreiðanlega að virka sannanlega rétt.
// Alls ekki má breyta þessu falli.  Athugið að
// þetta fall skilgreinir í raun þá virkni sem
// Partition og Sort eiga að hafa, þ.e. forskilyrði
// og eftirskilyrði þeirra falla.
method Test( m: multiset<int> )
{
   var s := Sort(m);
   assert IsSorted(s);
   assert m == multiset(s);
   if |m| > 0
   {
       var a,p,b := Partition(m);
       assert m == a+multiset{p}+b;
       assert forall z | z in a :: z<=p;
       assert forall z | z in b :: z>=p;
   }
}

// Aðalforritið er óþarfi, en er sett hér til gamans
// svo hægt sé að keyra eitthvað.
method Main()
{
   var x := Sort(multiset{0,9,1,8,2,7,3,6,4,5
                         ,0,9,1,8,2,7,3,6,4,5
                         }
                );
   print x;
}

///////////////////////////////////////////////////////////////
// Hér lýkur óbreytanlega hluta skrárinnar.
// Hér fyrir aftan er sá hluti sem þið eigið að breyta til að
// útfæra afbrigði af quicksort.
///////////////////////////////////////////////////////////////

method Partition( a: multiset<int> ) returns ( b: multiset<int>, p: int,
c: multiset<int> )
   // Bætið við requires/ensures eftir þörfum
   requires |a| >= 1;
```

Alexander Guðmundsson
alg35@hi.is

```
    ensures a == b+multiset{p}+c;
    ensures forall z | z in b :: z <= p;
    ensures forall z | z in c :: z >= p;
    ensures |b|==|b|<|a|;
    ensures |c|<|a|;
{
    // Forritið stofn fallsins.
    // Þið munið vilja nota lykkju.
    // Hjálparfallið RemoveOne verður væntanlega gagnlegt.
    var rest := a;
    p :| p in rest;
    rest := rest - multiset{p};
    b := multiset{};
    c := multiset{};
    while rest != multiset{}
        decreases |rest|;
        invariant a == rest+b+multiset{p}+c;
        invariant forall z | z in b :: z<=p;
        invariant forall z | z in c :: z>=p;
    {
        var z :| z in rest;
        if z <= p
        {
            b := b+multiset{z};
        }
        else
        {
            c := c+multiset{z};
        }
        rest := rest - multiset{z};
    }
}

method Sort( m: multiset<int> ) returns ( r: seq<int> )
    decreases m;
    ensures m == multiset(r);
    ensures IsSorted(r);
{
    // Forritið stofn fallsins.
    // Þið munið vilja nota endurkvæmni.
```

Alexander Guðmundsson
alg35@hi.is

```
    // Hjálparsetningin LomutoLemma
    // verður væntanlega gagnleg.
    // Hugsanlega viljið þið einnig
    // nota hjálparsetninguna Singleton.
    if |m| == 0
    {
        return [];
    }
    var b,p,c := Partition(m);
    var b' := Sort(b);
    var c' := Sort(c);
    r :=  b'+[p]+c';
    LomutoLemma(b,b',p,c,c',r);





}
```