// Höfundur spurningar:  Snorri Agnarsson, snorri@hi.is

// Permalink spurningar: https://rise4fun.com/Dafny/BnuhE

// Höfundur lausnar:     Alexander Guðmundsson

// Permalink lausnar:    https://rise4fun.com/Dafny/slSbJ

///////////////////////////////////////////////////////////////

// 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 í vaxandi röð.

predicate IsSorted( a: seq<int> )

{

    forall p,q | 0 <= p < q < |a| :: a[p] <= a[q]

}

// IsSegmented(a,b) er satt þá og því aðeins að

// sannað sé að öll gildi í a séu <= öll gildi í b.

predicate IsSegmented( a: seq<int> , b: seq<int> )

{

    (forall z,w | z in a && w in b :: z <= w) &&

    (forall p,q | 0 <= p < |a| && 0 <= q < |b| :: a[p] <= b[q])

}

// SortedEquals(a,b) sannar, fyrir raðaðar runur

// a og b, sem innihalda sama poka gilda, að runurnar

// eru jafnar.

lemma SortedEquals( a: seq<int>, b: seq<int> )

    requires multiset(a) == multiset(b);

    requires IsSorted(a);

    requires IsSorted(b);

    ensures a == b;

{

    if a == []

    {

        assert |b| == 0 || b[0] in multiset(a);

        return;

    }

    if b == []

    {

        assert |a| == 0 || a[0] in multiset(b);

        return;

    }

    assert a[0] in multiset(b);

    assert b[0] in multiset(a);

    assert a == a[0..1]+a[1..];

    assert b == b[0..1]+b[1..];

    assert a[0] == b[0];

    assert multiset(a[1..]) == multiset(a)-multiset{a[0]};

    assert multiset(b[1..]) == multiset(b)-multiset{b[0]};

    SortedEquals(a[1..],b[1..]);

}

// Samröðunarfall sem nota má í röksemdafærslu

// en ekki í raunverulegum útreikningum.

function MergeFun( a: seq<int>, b: seq<int> ): seq<int>

    decreases |a|+|b|;

{

    if a == [] then

        b

    else if b == [] then

        a

    else if a[0] <= b[0] then

        [a[0]]+MergeFun(a[1..],b)

    else

        [b[0]]+MergeFun(a,b[1..])

}

// Sannar að MergeFun(a,b) skilar réttu gildi.

// Fyrir mannlega lesendur er það augljóst,

// en Dafny þarf smá hjálp til að sanna það

// með þrepasönnun. Þið munuð vilja kalla á

// þessa hjálparsetningu ef þið byggið ykkar

// samröðun á endurkvæmni.

lemma MergeFunWorks( a: seq<int>, b: seq<int>, c: seq<int> )

    decreases |a|+|b|;

    requires IsSorted(a);

    requires IsSorted(b);

    requires c == MergeFun(a,b);

    ensures multiset(c) == multiset(a)+multiset(b);

    ensures IsSorted(c);

    ensures a!=[] && b!=[] && a[0]<=b[0] ==> c==a[0..1]+MergeFun(a[1..],b);

    ensures a!=[] && b!=[] && a[0]>=b[0] ==> c==b[0..1]+MergeFun(a,b[1..]);

{

    if a == [] || b == [] { return; }

    if a[0] <= b[0]

    {

        MergeFunWorks(a[1..],b,c[1..]);

        calc ==

        {

            multiset(c);

            multiset(c[0..1]+c[1..]);

            multiset(c[0..1])+multiset(c[1..]);

            multiset(c[0..1])+multiset(a[1..])+multiset(b);

            multiset(a[0..1])+multiset(a[1..])+multiset(b);

            multiset(a[0..1]+a[1..])+multiset(b);

            assert a[0..1]+a[1..] == a;

            multiset(a)+multiset(b);

        }

    }

    else

    {

        MergeFunWorks(a,b[1..],c[1..]);

        calc ==

        {

            multiset(c);

            multiset(c[0..1]+c[1..]);

            multiset(c[0..1])+multiset(c[1..]);

            multiset(c[0..1])+multiset(a)+multiset(b[1..]);

            multiset(b[0..1])+multiset(a)+multiset(b[1..]);

            multiset(a)+multiset(b[0..1])+multiset(b[1..]);

            multiset(a)+multiset(b[0..1]+b[1..]);

            assert b[0..1]+b[1..] == b;

            multiset(a)+multiset(b);

        }

    }

}

// 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);

{}

// Fjarlægir tvö gildi úr poka.  Getur verið gagnlegt

// í Split fallinu.

//

// Notkun: var b,x,y := RemoveTwo(a);

// Fyrir:  pokinn a inniheldur a.m.k. tvö gildi.

// Eftir:  a == b+multiset{x,y}

method RemoveTwo( a: multiset<int> ) returns( b: multiset<int>, x: int, y: int )

    requires |a| >= 2;

    ensures a == b+multiset{x,y};

{

    b := a;

    x :| x in b;

    b := b-multiset{x};

    assert |b| >= 1;

    y :| y in b;

    b := b-multiset{y};

}

// Prófunarfall sem staðfestir að Split og Sort

// séu áreiðanlega að virka sannanlega rétt.

// Alls ekki má breyta þessu falli.  Athugið að

// þetta fall skilgreinir í raun þá virkni sem

// Split og Sort eiga að hafa, þ.e. forskilyrði

// og eftirskilyrði þeirra falla.

method Test( x: multiset<int> )

{

    var a,b := Split(x);

    assert a+b == x;

    assert (|a|==|b|) || (|a|==|b|+1);

    a,b := Split(x);

    assert a+b == x;

    assert (|a|==|b|) || (|a|==|b|+1);

    var c := Sort(x);

    assert multiset(c) == x;

    assert IsSorted(c);

}

// 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 merge sort.

///////////////////////////////////////////////////////////////

// Þið munuð vilja nota þetta samröðunarfall í Sort fallinu.

method Merge( a: seq<int>, b: seq<int> ) returns( c: seq<int> )

    decreases a,b;

    requires IsSorted(a);

    requires IsSorted(b);

    ensures IsSorted(c);

    ensures multiset(a)+multiset(b) == multiset(c);

    ensures c == MergeFun(a,b);

{

    // Forritið stofn fyrir þetta fall

    // Þið getið notað lykkju eða endurkvæmni.

    // Sé endurkvæmni notuð þarf e.t.v. að bæta

    // við 'decreases' klausu í haus fallsins.

    //

    // Athugið að þið munuð næstum áreiðanlega

    // þurfa að kalla á hjálparsetninguna

    // MergeFunWorks á viðeigandi stöðum í

    // stofni fallsins.

    //

    // Ef þið notið lykkju þá er hugsanlegt að

    // hjálparsetningin SortedEquals verði

    // gagnleg.

    //

    // Einfaldara er að nota endurkvæmni en að

    // nota lykkju.  Munið að kalla á hjálpar-

    // setningar á viðeigandi stöðum til að

    // Dafny geti sannreynt það ástand sem

    // búið er að skapa.

    if a == []

    {

        return b;

    }

    if b == []

    {

        return a;

    }

    if a[0] > b[0]

    {

        c := Merge(a,b[1..]);

        c := b[0..1]+c;

    }

    else

    {

        c := Merge(a[1..],b);

        c := a[0..1]+c;

    }

    // Ráðlegt er að láta þessi tvö köll á

    // hjálparsetningar vera það síðasta sem gerist

    // í fallinu, sérstaklega ef þið notið lykkju.

    // Ef þið notið lykkju er einfaldara að ferðast

    // gegnum a og b frá vinstri til hægri.

    MergeFunWorks(a,b,MergeFun(a,b));

    SortedEquals(c,MergeFun(a,b));

}

// Skiptir innihaldi poka í tvennt þannig að pokarnir

// sem koma út eru nokkurn veginn jafn stórir.

method Split( a: multiset<int> )

        returns ( b: multiset<int>

                , c: multiset<int>

                )

    decreases a;

    ensures |b|==|c| || |b|==|c|+1;

    // Tilgreinið viðeigandi eftirskilyrði

    // fyrir þetta fall.

{

    if |a| == 1

    {

        var x :| x in a;

        b := b + multiset{x};

        return b,c;

    }

    if |a| == 0

    {

        return b,c;

    }

    // Forritið stofn fyrir þetta fall.

    // Þið getið notað lykkju eða endurkvæmni.

    // Sé endurkvæmni notuð þarf e.t.v. að bæta

    // við 'decreases' klausu í haus fallsins.

    //

    // Fallið RemoveTwo er gagnlegt hér.

    var d,e,f := RemoveTwo(a);

    b := b + multiset{e};

    c := c + multiset{f};

    b,c := Split(d);

}

// Raðar innihaldi poka yfir í runu með mergesort.

method Sort( a: multiset<int> ) returns ( b: seq<int> )

    // Tilgreinið viðeigandi 'decreases' og 'ensures'

    // klausur.

    decreases a;

    ensures IsSorted(b);

    ensures a == multiset(b);

{

    // Forritið stofn fyrir þetta fall.

    // Eðlilegt er að nota endurkvæmni hér.

    if |a| == 1

    {

        var x :| x in a;

        return [x];

    }

    var e,f := Split(a);

    if |c| < 2

    {

        var i :| i in c;

        var j :| j in d;

        b := Merge([i],[j]);

    }

    else

    {

        b := Sort(c);

        b := Sort(d);

    }

}