```java
// Höfundur: Snorri Agnarsson, snorri@hi.is

// Notið Link.java, sem er í Canvas, sem hjálparklasa.

// Vistið þessa skrá undir nafninu H12.java og gerið
// viðeigandi viðbætur þar sem þið finnið ???.

public class H12
{
    // Notkun: split(chain,w);
    // Fyrir:  chain er keðja með a.m.k. tvo hlekki.
    //         w er tveggja staka Link<T>[], þ.e. w.length == 2.
    // Eftir:  w[0] og w[1] eru keðjur sem samanlagt innihalda
    //         hlekkina úr upphaflega chain, í einhverri röð.
    //         Fjöldi hlekkja í w[0] er annaðhvort sami og í w[1]
    //         eða einum meiri.
    //         Athugið að ekki má úthluta neinum nýjum hlekkjum
    //         og reyndar ekki neinum nýjum minnissvæðum.
    public static<T extends Comparable<? super T>>
    void split( Link<T> chain, Link<T>[] w )
    {
        // Hér vantar forritstexta.
        // Notið lykkju.  Aðalatriðið hér er að fastayrðingin
        // lykkjunnar sé góð. Ekki fást mörg stig fyrir lausn
        // sem ekki hefur góða fastayrðingu jafnvel þótt
        // fallið virki samkvæmt lýsingu.
        // Þægilegt er að nota tvö break til að komast út úr
        // lykkjunni, í öðru tilvikinu þegar búið er að skipta
        // chain jafnt milli tveggja keðja og í hinu tiltikinu
        // þegar munar einum á fjölda hlekkja.
        // Reyndar er fleiri en ein þægileg leið til að leysa
        // þetta í lykkju.
        Link<T> y = new Link<T>();
        Link<T> z = new Link<T>();
        Link<T> temp = new Link<T>();


        while(chain != null)

            //  y inniheldur einum fleiri eða jafn marga hlekki
            //  og z,
            //  chain þarf að innihalda að minnasta kosti einn hlekk
            //  þá verður y með einum fleiri hlekki en z,
            //  allir hlekkir í z og y eru í gamla chain
```

```
    {


        if(chain.head == null || chain.tail == null) {
            break;
        }
        temp = chain.tail;
        chain.tail = y;
        y = chain;
        chain = temp;
        if(chain.head == null || chain.tail == null) {
            break;
        }
        temp = chain.tail;
        chain.tail = z;
        z = chain;
        chain = temp;
    }
    w[0] = y;
    w[1] = z;
}



// Notkun: Link<T> y = mergeSort(x,w);
// Fyrir:  x er lögleg keðja þar sem hlekkirnir innihalda
//         lögleg gildi af tagi T.
//         w er tveggja staka Link<T>[], þ.e. w.length == 2.
// Eftir:  y er keðja sömu hlekkja þannig að hlekkirnir
//         í y eru í vaxandi hausaröð miðað við compareTo
//         fyrir hluti af tagi T.
//         Fylkið w inniheldur engin sérstök skilgreind
//         gildi.
//         Athugið að ekki má úthluta neinum nýjum hlekkjum
//         og reyndar ekki neinum nýjum minnissvæðum.
public static<T extends Comparable<? super T>>
Link<T> mergeSort( Link<T> x, Link<T>[] w )
{
    Link<T> y = new Link<T>();
    Link<T> z = new Link<T>();
    // Hér vantar forritstexta.
    if(x.head != null || x.tail.head != null) {
        split(x,w);
```

```
            y = w[0];
            z = w[1];
            y = mergeSort(y,w);
            z = mergeSort(z,w);
            y = merge(y,z);

        }
        else {
            return x;
        }
            return y;

    }

    // Notkun: Link<T> z = merge(x,y);
    // Fyrir:  x og y eru ekki-tómar keðjur í vaxandi röð með
    //         enga sameiginlega hlekki.
    // Eftir:  z er keðja í vaxandi röð sem inniheldur
    //         alla hlekkina úr x og y og enga aðra.
    //         Athugið að ekki má úthluta neinum nýjum hlekkjum
    //         og reyndar ekki neinum nýjum minnissvæðum.
    public static<T extends Comparable<? super T>>
    Link<T> merge( Link<T> x, Link<T> y )
    {
        // Hér vantar forritstexta.
        // Notið lykkju.  Aðalatriðið hér er að fastayrðingin
        // lykkjunnar sé góð. Ekki fást mörg stig fyrir lausn
        // sem ekki hefur góða fastayrðingu jafnvel þótt
        // fallið virki samkvæmt lýsingu.
        // Athugið að þægilegt er að byrja á að frumstilla
        // ekki tóma útkomukeðju með því að nota fremsta
        // hlekk úr x eða y, eftir því hvor inniheldur minna
        // gildi. Þægilegt er einnig að viðhalda því að
        // einhver breyta vísi á aftasta hlekk þeirrar keðju.
        Link<T> z = new Link<T>();
        Link<T> w = new Link<T>();


        if(x.head.compareTo(y.head) < 0) {
            z = x;
            w = z;
            x = x.tail;
        }
        else {
            z = y;
```

```
            w = z;
            y = y.tail;
        }

        //  z er keðja í vandi röð,
        //  hvert hlekkur í z er minni hlekkurinn af
        //  x og y meðan við compareTo
        while (x != null && y!= null) {
            if(x.head.compareTo(y.head) < 0) {
                w.tail = x;
                w = x;
                x = x.tail;
            }
            else {
                w.tail = y;
                w = y;
                y = y.tail;
            }
        }
        w.tail = null;
        return z;
    }

    // Notkun: Link<T> x = makeChain(a,i,j);
    // Fyrir:  a er T[], ekki null.
    //         0 <= i <= j <= a.length.
    // Eftir:  x vísar á keðju nýrra hlekkja sem innihalda
    //         gildin a[i..j), í þeirri röð, sem hausa.
    public static<T> Link<T> makeChain( T[] a, int i, int j )
    {
        if( i==j ) return null;
        Link<T> x = new Link<T>();
        x.head = a[i];
        x.tail = makeChain(a,i+1,j);
        return x;
    }

    // Keyrið skipanirnar
    //   javac H12.java
    //   java H12 1 2 3 4 3 2 1 10 30 20
    // og sýnið útkomuna í athugasemd hér:
         //Stack Overflow Error
    public static void main( String[] args )
    {
        Link<String> x = makeChain(args,0,args.length);
```

```
        Link<String>[] w = (Link<String>[])new Link<?>[2];
        x = mergeSort(x,w);
        while( x != null )
        {
            System.out.print(x.head+" ");
            x = x.tail;
        }
    }
}
```