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

// Listar með hliðarverkunum í Java.

// Höfundur lausnar: Alexander Guðmundsson, alg35@hi.is

public class H9

{

    // Tilvik af link eru breytanlegir hlekkir með

    // haus sem er heiltala og hala sem er endanleg

    // keðja hlekkja.  Tóm keðja er táknuð með null.

    // Það er mögulegt að búa til hringkeðjur og það

    // er mögulegt að breyta bæði haus og hala.

    public static class Link

    {

        public int head;

        public Link tail;

    }

    // Notkun: H9.Link x = H9.cons(head,tail);

    // Fyrir:  head er heiltala, tail er H9.Link (má vera null).

    // Eftir:  x er tilvísun á nýjan H9.Link með gefinn haus og

    //         og hala.

    public static Link cons( int h, Link t )

    {

        Link newLink = new Link();

        newLink.head = h;

        newLink.tail = t;

        return newLink;

    }

    // Notkun: int n = H9.length(x);

    // Fyrir:  x er H9.Link tilvísun, má vera null

    //         en má ekki vísa á hringkeðju.

    // Eftir:  n er fjöldi hlekkja í keðju x.

    public static int length( H9.Link x )

    {

        if(x == null) {

            return 0;

        }

        int n = 1;

        Link tail = x.tail;

        //  Skref:  tail halinn verður hali tail halans

        //          n stækkar um 1

        //  Eftir:  n er jafnt dýptinni á x

        while(tail != null) {

            tail = tail.tail;

            n += 1;

        }

        return n;

    }

    // Notkun: int i = H9.nth(x,n);

    // Fyrir:  x er keðja með a.m.k. n+1 hlekki.

    // Eftir:  i er hausinn á n-ta hlekk í keðjunni

    //         þar sem 0-ti hlekkur er fremsti hlekkur.

    public static int nth( H9.Link x, int n )

    {

        int i = 0;

        //  Skref:  x er jafn halanum af x

        //          i hækkar um 1

        //  Eftir:  i er jafnt n

        //          hausinn á x er gildi númer n á

        //          upprunarlega x

        while(i != n) {

            x = x.tail;

            i += 1;

        }

        return x.head;

    }

    // Notkun: H9.Link x = makeChain(a);

    // Fyrir:  a er tilvísun á int[]. Má ekki vera null

    //         en má vera tómt.

    // Eftir:  x er keðja sem inniheldur gildin í a

    //         þannig að fyrir i=0,...,a.length gildir

    //         H9.nth(x,i) == a[i].

    public static Link makeChain( int[] a )

    {

        int head = a[a.length-1];

        Link x = cons(head,null);

        //  Skref:  hali x verður að x

        //          haus x verður að a[i]

        //  Eftir:  x verður að Link af a.

        for(int i = a.length-2; i >= 0; i--) {

            x = cons(a[i],x);

        }

        return x;

    }

    // Notkun: int i = H9.last(x);

    // Fyrir:  x er tilvísun á H9.Link, má ekki vera null

    //         og má ekki vera hringkeðja.

    // Eftir:  i er gildið í (hausinn á) aftasta hlekk x.

    public static int last( Link x )

    {

        //  Skref:  x verður að hala x

        //  Eftir:  aðeins haus eftir í x sem er

        //          síðasta gildið

        while(x.tail != null) {

            x = x.tail;

        }

        return x.head;

    }

    // Notkun: H9.Link z = H9.destructiveRemoveLast(x);

    // Fyrir:  x er tilvísun á H9.Link, má ekki vera null

    //         og má ekki vera hringkeðja.

    // Eftir:  z er keðja sem inniheldur sömu hlekki í

    //         sömu röð og x, nema hvað hlekkurinn sem

    //         var aftast er ekki lengur í keðjunni og

    //         í stað tilvísunar á þann hlekk inniheldur nú

    //         aftasti hlekkurinn hala sem er null.

    //         Eftir kallið eru sömu heiltölugildi í

    //         hlekkjunum og sömu halar, fyrir utan í

    //         hlekknum sem nú er aftast (ef einhver er).

    //         Gilda þarf að E9.length(z) == gamla(E9.length(x))-1

    //         og fyrir i=0,...,E9.length(z)-1 þarf að gilda

    //         E9.nth(z,i) == gamla(E9.nth(x,i)).

    public static Link destructiveRemoveLast( Link x )

    {

        int length = length(x);

        int nth = nth(x,length-1);

        Link z = cons(nth,null);

        length = length - 1;

        //  Skref:  nth verður gildi númer length-1

        //          hali af z verður að z

        //          haus af z verður að nth

        //          length minkar um 1

        //  Eftir:  lengd er orðin 0

        //          lengd á z er lengd á x - 1

        //          z er x án seinasta hala

        while(length > 0) {

            nth = nth(x,length-1);

            z = cons(nth,z);

            length = length - 1;

        }

        return z;

    }

    // Notkun: H9.Link r = H9.destructiveReverse(x);

    // Fyrir:  x er keðja, má vera tóm.

    // Eftir:  z er keðja sömu hlekkja og x, þannig að

    //         hlekkirnir í r eru í öfugri röð miðað

    //         við gamla x. Heiltölugildin í hlekkjunum

    //         eru óbreytt.

    public static Link destructiveReverse( Link x )

    {

        Link y = null;

        //  Skref:  aftasta gildi x verður nýji haus y

        //          hali y verður gamla y

        //          aftasta gildi af keðju x er fjarlægt

        //  Eftir:  y er jafn djúpt og gamla x

        //          y er eins og öfug röðun af gamla x

        //          x er tómt

        while(x != null)

        {

            // y inniheldur keðju núll eða fleiri hlekkja

            // sem hafa verið fjarlægðir framan af x.

            // Röð hlekkjanna í y er öfug röð þeirra þegar

            // þeir voru í x.  Innihald hlekkjanna er

            // óbreytt fyrir utan breytingar á hala þeirra.

            int head = last(x);

            y = cons(head, y);

            x = destructiveRemoveLast(x);

        }

        return y;

    }

    // Keyrið skipunina

    //   java H9 1 2 3 4

    // og sýnið hvað forritið skrifar

    public static void main( String[] args )

    {

        H9.Link x = null;

        for( int i=0 ; i!=args.length ; i++ )

            x = H9.cons(Integer.parseInt(args[i]),x);

        while( x != null )

        {

            H9.Link z = H9.destructiveReverse(x);

            x = z;

            while( z != null )

            {

                System.out.print(z.head); System.out.print(" ");

                z = z.tail;

            }

            x = H9.destructiveRemoveLast(x);

            System.out.println();

        }

    }

}

