/\*

\* To change this template, choose Tools | Templates

\* and open the template in the editor.

\*/

/\*\*

\* this class Cons2 implements a Lisp-like Cons2 cell

\*

\* @author Gordon S. Novak Jr.

\* @version 29 Nov 01; 25 Aug 08; 05 Sep 08; 08 Sep 08; 02 Sep 09; 27 Jan 10

\* 05 Feb 10; 16 Jul 10; 02 Sep 10; 13 Jul 11

\*/

import java.util.\*;

public class Cons2

{

// instance variables

private Object car;

private Cons2 cdr;

private Cons2(Object first, Cons2 rest)

{ car = first;

cdr = rest; }

// make a new Cons2 and put the arguments into it

// add one new thing to the front of an existing list

// Cons2("a", list("b", "c")) = (a b c)

public static Cons2 Cons2(Object first, Cons2 rest)

{ return new Cons2(first, rest); }

// test whether argument is a Cons2

public static boolean Cons2p (Object x)

{ return ( (x != null) && (x instanceof Cons2) ); }

// first thing in a list: first(list("a", "b", "c")) = "a"

// safe, returns null if lst is null

public static Object first(Cons2 lst) {

return ( (lst == null) ? null : lst.car ); }

// rest of a list after the first thing:

// rest(list("a", "b", "c")) = (b c)

// safe, returns null if lst is null

public static Cons2 rest(Cons2 lst) {

return ( (lst == null) ? null : lst.cdr ); }

// second thing in a list: second(list("a", "b", "c")) = "b"

public static Object second (Cons2 x) { return first(rest(x)); }

// third thing in a list: third(list("a", "b", "c")) = "c"

public static Object third (Cons2 x) { return first(rest(rest(x))); }

// destructively replace the first

public static void setfirst (Cons2 x, Object i) { x.car = i; }

// destructively replace the rest

public static void setrest (Cons2 x, Cons2 y) { x.cdr = y; }

// make a list of things: list("a", "b", "c") = (a b c)

public static Cons2 list(Object ... elements) {

Cons2 list = null;

for (int i = elements.length-1; i >= 0; i--) {

list = Cons2(elements[i], list);

}

return list;

}

// convert a list to a string for printing

public String toString() {

return ( "(" + toStringb(this) ); }

public static String toString(Cons2 lst) {

return ( "(" + toStringb(lst) ); }

private static String toStringb(Cons2 lst) {

return ( (lst == null) ? ")"

: ( first(lst) == null ? "()" : first(lst).toString() )

+ ((rest(lst) == null) ? ")"

: " " + toStringb(rest(lst)) ) ); }

public static int square(int x) { return x\*x; }

// \*\*\*\*\*\* your code starts here \*\*\*\*\*\*

// Sum of squares of integers from 1..n

public static int sumsq (int n) {

if (n<=0)

return 1;

else

return square(n) + sumsq(n-1);

}

// Addition using Peano arithmetic

public static int peanoplus(int x, int y) {

if (y==0)

return x;

else

return peanoplus(x+1,y-1);

}

// Multiplication using Peano arithmetic

public static int peanotimes(int x, int y) {

if (y==1)

return x;

else

return x + peanotimes(x, y-1);

}

// n choose k: distinct subsets of k items chosen from n items

public static int choose(int n, int k) {

int control = k + 1;

int answer = 1;

return chooseb( n, 1, control, answer);

}

public static int chooseb (int n, int k, int ctrl, int ans) {

if (k == ctrl)

return 1;

else

{

ans = n \* chooseb(n-1, k+1, ctrl, ans);

return ans/k;

}

}

// Add up a list of Integer

// iterative version, using while

public static int sumlist (Cons2 lst) {

int sum = 0;

while ( lst != null ) {

sum += (Integer) first(lst); // cast since first() can be Object

lst = rest(lst); }

return sum; }

// second iterative version, using for

public static int sumlistb (Cons2 arg) {

int sum = 0;

for (Cons2 lst = arg ; lst != null; lst = rest(lst) )

sum += (Integer) first(lst); // cast since first() can be Object

return sum; }

// recursive version

public static int sumlistr (Cons2 lst) {

if (lst == null)

return 0;

else

return (Integer)first(lst) +sumlistr(rest(lst));

}

// tail recursive version

public static int sumlisttr (Cons2 lst) {

return sumlistr(lst);

}

// Sum of squared differences of elements of two lists

// iterative version

public static int sumsqdiff (Cons2 lst, Cons2 lstb) {

int sum = 0;

while(lst != null && lstb != null)

{

sum += Math.pow((Integer)first(lst) - (Integer)first(lstb),2);

lst = rest(lst);

lstb = rest(lstb);

}

return sum;

}

// recursive version

public static int sumsqdiffr (Cons2 lst, Cons2 lstb) {

int sentence;

if (lst == null || lstb == null)

return 0;

else

{

sentence = ((Integer)first(lst) - (Integer)first(lstb)) \*

((Integer)first(lst) - (Integer)first(lstb));

return sentence + sumsqdiffr(rest(lst),rest(lstb));

}

}

// tail recursive version

public static int sumsqdifftr (Cons2 lst, Cons2 lstb) {

return sumsqdiffr(lst,lstb);

}

// Find the maximum value in a list of Integer

// iterative version

public static int maxlist (Cons2 lst) {

int max = (Integer)first(lst);

for (Cons2 l1 = lst ; l1 != null ; l1 = rest(l1))

{

if (max < (Integer)first(l1))

max = (Integer)first(l1);

}

return max;

}

// recursive version

public static int maxlistr (Cons2 lst) {

if (lst==null)

return 0;

else

return Math.max((Integer)first(lst),maxlistr(rest(lst)));

}

// tail recursive version

public static int maxlisttr (Cons2 lst) {

return maxlistr(lst);

}

// Make a list of Binomial coefficients

// binomial(2) = (1 2 1)

public static Cons2 binomial(int n) {

Cons2 lst = null;

lst = Cons2(1,lst);

for (int i = 0 ; i <= n ; i++)

{

lst = Cons2(choose(n,i),lst);

}

return lst;

}

// \*\*\*\*\*\* your code ends here \*\*\*\*\*\*

public static void main( String[] args )

{

System.out.println("sumsq(5) = " + sumsq(5));

System.out.println("peanoplus(3, 5) = " + peanoplus(3, 5));

System.out.println("peanotimes(3, 5) = " + peanotimes(3, 5));

System.out.println("peanotimes(30, 30) = " + peanotimes(30, 30));

System.out.println("choose 5 3 = " + choose(5, 3));

System.out.println("choose 100 3 = " + choose(100, 3));

System.out.println("choose 20 10 = " + choose(20, 10));

System.out.println("choose 100 5 = " + choose(100, 5));

for (int i = 0; i <= 4; i++)

System.out.println("choose 4 " + i + " = " + choose(4, i));

Cons2 mylist = list(Integer.valueOf(3), Integer.valueOf(4),

Integer.valueOf(8), Integer.valueOf(2));

Cons2 mylistb = list(Integer.valueOf(2), Integer.valueOf(1),

Integer.valueOf(6), Integer.valueOf(5));

System.out.println("mylist = " + mylist);

System.out.println("sumlist = " + sumlist(mylist));

System.out.println("sumlistb = " + sumlistb(mylist));

System.out.println("sumlistr = " + sumlistr(mylist));

System.out.println("sumlisttr = " + sumlisttr(mylist));

System.out.println("mylistb = " + mylistb);

System.out.println("sumsqdiff = " + sumsqdiff(mylist, mylistb));

System.out.println("sumsqdiffr = " + sumsqdiffr(mylist, mylistb));

System.out.println("sumsqdifftr = " + sumsqdifftr(mylist, mylistb));

System.out.println("maxlist " + mylist + " = " + maxlist(mylist));

System.out.println("maxlistr = " + maxlistr(mylist));

System.out.println("maxlisttr = " + maxlisttr(mylist));

//System.out.println("binomial(4) = " + binomial(4));

//System.out.println("binomial(20) = " + binomial(20));

}

}