/\*\*

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

\*

\* **@author** Gordon S. Novak Jr.

\* **@version** 29 Nov 01; 25 Aug 08; 05 Sep 08; 08 Sep 08; 12 Sep 08; 24 Sep 08

\* 02 Oct 09; 12 Feb 10; 04 Oct 12

\*/

**interface** Functor { Object fn(Object x); }

**interface** Predicate { **boolean** pred(Object x); }

**public** **class** Cons4

{

// instance variables

**private** Object car;

**private** Cons4 cdr;

**private** Cons4(Object first, Cons4 rest)

{ car = first;

cdr = rest; }

**public** **static** Cons4 cons(Object first, Cons4 rest)

{ **return** **new** Cons4(first, rest); }

**public** **static** **boolean** consp (Object x)

{ **return** ( (x != **null**) && (x **instanceof** Cons4) ); }

// safe car, returns null if lst is null

**public** **static** Object first(Cons4 lst) {

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

// safe cdr, returns null if lst is null

**public** **static** Cons4 rest(Cons4 lst) {

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

**public** **static** Object second (Cons4 x) { **return** *first*(*rest*(x)); }

**public** **static** Object third (Cons4 x) { **return** *first*(*rest*(*rest*(x))); }

**public** **static** **void** setfirst (Cons4 x, Object i) { x.car = i; }

**public** **static** **void** setrest (Cons4 x, Cons4 y) { x.cdr = y; }

**public** **static** Cons4 list(Object ... elements) {

Cons4 list = **null**;

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

list = *cons*(elements[i], list);

}

**return** list;

}

// convert a list to a string for printing

**public** String toString() {

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

**public** **static** String toString(Cons4 lst) {

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

**private** **static** String toStringb(Cons4 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; }

// iterative destructive merge using compareTo

**public** **static** Cons4 dmerj (Cons4 x, Cons4 y) {

**if** ( x == **null** ) **return** y;

**else** **if** ( y == **null** ) **return** x;

**else** { Cons4 front = x;

**if** ( ((Comparable) *first*(x)).compareTo(*first*(y)) < 0)

x = *rest*(x);

**else** { front = y;

y = *rest*(y); };

Cons4 end = front;

**while** ( x != **null** )

{ **if** ( y == **null** ||

((Comparable) *first*(x)).compareTo(*first*(y)) < 0)

{ *setrest*(end, x);

x = *rest*(x); }

**else** { *setrest*(end, y);

y = *rest*(y); };

end = *rest*(end); }

*setrest*(end, y);

**return** front; } }

**public** **static** Cons4 midpoint (Cons4 lst) {

Cons4 current = lst;

Cons4 prev = current;

**while** ( lst != **null** && *rest*(lst) != **null**) {

lst = *rest*(*rest*(lst));

prev = current;

current = *rest*(current); };

**return** prev; }

// Destructive merge sort of a linked list, Ascending order.

// Assumes that each list element implements the Comparable interface.

// This function will rearrange the order (but not location)

// of list elements. Therefore, you must save the result of

// this function as the pointer to the new head of the list, e.g.

// mylist = llmergesort(mylist);

**public** **static** Cons4 llmergesort (Cons4 lst) {

**if** ( lst == **null** || *rest*(lst) == **null**)

**return** lst;

**else** { Cons4 mid = *midpoint*(lst);

Cons4 half = *rest*(mid);

*setrest*(mid, **null**);

**return** *dmerj*( *llmergesort*(lst),

*llmergesort*(half)); } }

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

// add other functions as you wish.

/// Merge algorithm was used to create union function

**public** **static** Cons4 union (Cons4 x, Cons4 y) {

Cons4 sortedX = *llmergesort*(x);

Cons4 sortedY = *llmergesort*(y);

**if** (sortedX == **null**)

**return** sortedY;

**else** **if** (sortedY == **null**)

**return** sortedX;

**else** **if** ( ((Comparable) *first*(sortedX))

.compareTo(*first*(sortedY)) < 0 )

**return** *cons*(*first*(sortedX),

*union*(*rest*(sortedX), sortedY));

**else** **if** ( ((Comparable) *first*(sortedX))

.compareTo(*first*(sortedY)) == 0 )

**return** *union* (*rest*(sortedX), sortedY);

**else**

**return** *cons*(*first*(sortedY),

*union*(sortedX, *rest*(sortedY)));

}

// union(x, y)

// sort(x), sort(y)

// return merge(x, y)

// merge(x, y)

**public** **static** Cons4 setDifference (Cons4 x, Cons4 y) {

Cons4 sortedX = *llmergesort*(x);

Cons4 sortedY = *llmergesort*(y);

**if** (sortedX == **null**)

**return** **null**;

**else** **if** (sortedY == **null**)

**return** sortedX;

**else** **if** ( ((Comparable) *first*(sortedX))

.compareTo(*first*(sortedY)) == 0 )

**return** *setDifference*(*rest*(sortedX), *rest*(sortedY));

**else**

**return** *cons*(*first*(sortedX),

*setDifference*(*rest*(sortedX), sortedY));

}

**public** **static** Cons4 bank(Cons4 accounts, Cons4 updates) {

Cons4 sortedUpdates = **null**;

**for** (Cons4 argi = updates ; argi != **null** ; argi = *rest*(argi))

**for** (Cons4 argj = argi ; argj != **null** ; argj = *rest*(argj))

**if** (((Account)*first*(argj)).compareTo((Account)*first*(*rest*(argj))) == -1)

{

Cons4 temp = argj;

*cons*(*first*(argj), sortedUpdates);

}

System.*out*.println("sortedUpdates is: " + sortedUpdates.toString());

**return** **null**;

}

**public** **static** String [] mergearr(String [] x, String [] y) {

**int** size = x.length + y.length - 1;

String [] answer = **new** String [size];

**int** k = 0;

**return** *mergearrb*( x, y, answer, k);

}

**public** **static** String [] mergearrb(String [] x, String [] y, String [] answer , **int** k) {

**if** (x.length == 0)

{

answer = *StringArrConnect*(answer,y);

**return** answer;

}

**else** **if** (y.length == 0)

{

answer = *StringArrConnect*(answer,x);

**return** answer;

}

**else** **if** (x[0].compareTo(y[0]) < 0 )

{

**int** size = x.length - 1;

String [] z = **new** String [size];

answer[k] = x[0];

k++;

**for** (**int** i = 0 ; i < size ; i++)

z[i] = x[i+1];

**return** *mergearrb* (z,y,answer,k);

}

**else**

{

**int** size = y.length - 1;

String [] z = **new** String [size];

answer[k] = y[0];

k++;

**for** (**int** i = 0 ; i < size ; i++)

z[i] = y[i+1];

**return** *mergearrb* (x,z,answer,k);

}

}

**public** **static** String [] StringArrConnect(String [] a , String [] b)

{

**int** k = 0;

**for** (**int** i = 0 ; i < a.length ; i++)

**if** (a[i] == "" )

{

a[i] = b[k];

k++;

}

**return** a;

}

**public** **static** **boolean** markup(Cons4 text) {

}

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

**public** **static** **void** main( String[] args )

{

Cons4 set1 = *list*("d", "b", "c", "a");

Cons4 set2 = *list*("f", "d", "b", "g", "h");

System.*out*.println("set1 = " + Cons4.*toString*(set1));

System.*out*.println("set2 = " + Cons4.*toString*(set2));

System.*out*.println("union = " + Cons4.*toString*(*union*(set1, set2)));

Cons4 set3 = *list*("d", "b", "c", "a");

Cons4 set4 = *list*("f", "d", "b", "g", "h");

System.*out*.println("set3 = " + Cons4.*toString*(set3));

System.*out*.println("set4 = " + Cons4.*toString*(set4));

System.*out*.println("difference = " +

Cons4.*toString*(*setDifference*(set3, set4)));

Cons4 accounts = *list*(

**new** Account("Arbiter", **new** Integer(498)),

**new** Account("Flintstone", **new** Integer(102)),

**new** Account("Foonly", **new** Integer(123)),

**new** Account("Kenobi", **new** Integer(373)),

**new** Account("Rubble", **new** Integer(514)),

**new** Account("Tirebiter", **new** Integer(752)),

**new** Account("Vader", **new** Integer(1024)) );

Cons4 updates = *list*(

**new** Account("Foonly", **new** Integer(100)),

**new** Account("Flintstone", **new** Integer(-10)),

**new** Account("Arbiter", **new** Integer(-600)),

**new** Account("Garble", **new** Integer(-100)),

**new** Account("Rabble", **new** Integer(100)),

**new** Account("Flintstone", **new** Integer(-20)),

**new** Account("Foonly", **new** Integer(10)),

**new** Account("Tirebiter", **new** Integer(-200)),

**new** Account("Flintstone", **new** Integer(10)),

**new** Account("Flintstone", **new** Integer(-120)) );

System.*out*.println("accounts = " + accounts.toString());

System.*out*.println("updates = " + updates.toString());

Cons4 newaccounts = *bank*(accounts, updates);

// System.out.println("result = " + newaccounts.toString());

String[] arra = {"a", "big", "dog", "hippo"};

String[] arrb = {"canary", "cat", "fox", "turtle"};

String[] resarr = *mergearr*(arra, arrb);

**for** ( **int** i = 0; i < resarr.length; i++ )

System.*out*.println(resarr[i] );

Cons4 xmla = *list*( "<TT>", "foo", "</TT>");

Cons4 xmlb = *list*( "<TABLE>", "<TR>", "<TD>", "foo", "</TD>",

"<TD>", "bar", "</TD>", "</TR>",

"<TR>", "<TD>", "fum", "</TD>", "<TD>",

"baz", "</TD>", "</TR>", "</TABLE>" );

Cons4 xmlc = *list*( "<TABLE>", "<TR>", "<TD>", "foo", "</TD>",

"<TD>", "bar", "</TD>", "</TR>",

"<TR>", "<TD>", "fum", "</TD>", "<TD>",

"baz", "</TD>", "</WHAT>", "</TABLE>" );

Cons4 xmld = *list*( "<TABLE>", "<TR>", "<TD>", "foo", "</TD>",

"<TD>", "bar", "</TD>", "", "</TR>",

"</TABLE>", "</NOW>" );

Cons4 xmle = *list*( "<THIS>", "<CANT>", "<BE>", "foo", "<RIGHT>" );

Cons4 xmlf = *list*( "<CATALOG>",

"<CD>",

"<TITLE>", "Empire", "Burlesque", "</TITLE>",

"<ARTIST>", "Bob", "Dylan", "</ARTIST>",

"<COUNTRY>", "USA", "</COUNTRY>",

"<COMPANY>", "Columbia", "</COMPANY>",

"<PRICE>", "10.90", "</PRICE>",

"<YEAR>", "1985", "</YEAR>",

"</CD>",

"<CD>",

"<TITLE>", "Hide", "your", "heart", "</TITLE>",

"<ARTIST>", "Bonnie", "Tyler", "</ARTIST>",

"<COUNTRY>", "UK", "</COUNTRY>",

"<COMPANY>", "CBS", "Records", "</COMPANY>",

"<PRICE>", "9.90", "</PRICE>",

"<YEAR>", "1988", "</YEAR>",

"</CD>", "</CATALOG>");

System.*out*.println("xmla = " + xmla.toString());

System.*out*.println("result = " + *markup*(xmla));

System.*out*.println("xmlb = " + xmlb.toString());

System.*out*.println("result = " + *markup*(xmlb));

System.*out*.println("xmlc = " + xmlc.toString());

System.*out*.println("result = " + *markup*(xmlc));

System.*out*.println("xmld = " + xmld.toString());

System.*out*.println("result = " + *markup*(xmld));

System.*out*.println("xmle = " + xmle.toString());

System.*out*.println("result = " + *markup*(xmle));

System.*out*.println("xmlf = " + xmlf.toString());

System.*out*.println("result = " + *markup*(xmlf));

}

}