

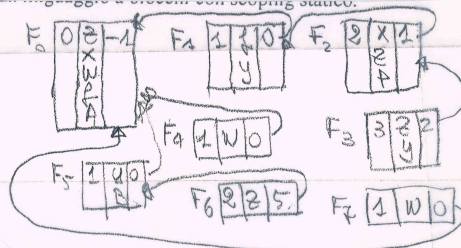
Esercizio 1 soluzione

```

(int z = 7; int x; int w; int y;
{int f(int x, int z){int x = x + w;
void p(int z){int y = x + z; z = f(y + w);}
return(y+x+w);}
int y = z+x;}
int f(int w){return(w+x+z);}
void A(int y){void B(int z){y=f(x+z);x=w}
{int w = z; A(f(x+z, y));}
A(f(z));}

```

Si modifica un campo statico per tale codice:



Esercizio 2 soluzione

- (a) let rec rev = fun l -> match l with
 [] -> []
 |x::lR -> List.append (rev lR) [x]
 ;;
 rev l;;
- (b) let rec revT = fun l r -> match l with
 [] -> r
 |x::lR -> revT lR (List.append [x] r)
 ;;
 revT l [];;
- (c) let revl = fun l -> List.fold_right (fun x -> fun u -> List.append u [x]) l [];;
 revl l;;

Esercizio 3 soluzione

- (a) public class Fun2<A,B> extends Fun<A,B>{
 public boolean inj(){
 LinkedList<A> d = dom();
 for(A x : d){
 for(A y : d){
 if(!x.equals(y) && apply(x).equals(apply(y))) return false;};
 return true;};
 }
- (b) public class Fun3<A,B> extends Fun<A,B>{
 LinkedList cod;
 public Fun3(){
 super();
 cod = new LinkedList();
 public void add(A x, B y){
 if (!cod.contains(y)) {
 if (isIn(x)){
 B oldBvalue = apply(x);
 if (y.equals(oldBvalue)) return;
 cod.remove(oldBvalue);
 super.add(x,y);
 cod.add(y);
 return;}
 return;}
 public void remove(A x){
 if (isIn(x)){
 cod.remove(apply(x));
 super.remove(x);}
 private boolean isIn(A x){
 return dom().contains(x);}
 }
- (c) public class Fun4<A,B> extends Fun<A,B>{
 public LinkedList im(Fun4<A,B> g){
 LinkedList codg = new LinkedList();
 for(A x : g.dom()) codg.add(g.apply(x));
 LinkedList res = new LinkedList();
 for(A x : dom()) {
 B iThis = apply(x);
 if(!codg.contains(iThis)) res.add(iThis);
 return res;}}