```
{
Autor : Slawek Kolasinski
Data  : 02.2009
e-mail: skola@mimuw.edu.pl
}

Program SLLists;
type
   typ     = integer;
   lista   = ^element;
   element = record
                dane : typ;
                ogon : lista;
             end;

{ Inicjuje liste }
procedure inicjuj(var l : lista);
begin
   l := nil;
end; { inicjuj }

{ Sprawdza czy lista jest pusta }
function pusta(l : lista) : boolean;
begin
   pusta := (l = nil);
end; { pusta }

{ Zwraca wartosc zapisana w pierwszym elemencie listy
  ale nie modyfikuje listy }
function glowa(l : lista) : typ;
begin
   if not pusta(l) then
      glowa := l^.dane
   else
   begin
      writeln('glowa: pusta lista!');
      halt;
   end;
end; { glowa }

{ Zwraca ogon listy, tzn. cala liste bez pierwszego elementu.
  Nie modyfikuje listy. }
function ogon(l : lista) : lista;
begin
   if not pusta(l) then
      ogon := l^.ogon
   else
   begin
      writeln('ogon: pusta lista!');
      halt;
   end;
end; { ogon }

{ Usuwa pierwszy element z listy }
function usun(var l : lista) : typ;
var
   czubek : lista;
begin
   usun := glowa(l);
   czubek := l;
   l := ogon(l);
   dispose(czubek);
end; { usun }

{ Dodaje na poczatek listy nowy element }
procedure dodaj(var l : lista; x : typ);
var
   elt : lista;
begin
   new(elt);
   elt^.dane := x;
   elt^.ogon := l;
   l := elt;
end; { dodaj }

{ Podlacza liste l2 na konicu listy l1.
  Lista l1 jest wydluzana o elementy listy l2.
  Lista l2 jest oprozniana. }
procedure polacz(var l1, l2 : lista);
var
```



```
   aktualny, poprzedni : lista;
begin
   if not pusta(l1) then
   begin
      poprzedni := nil;
      aktualny := l1;
      while not pusta(aktualny) do
      begin
         poprzedni := aktualny;
         aktualny := ogon(aktualny);
      end;
      poprzedni^.ogon := l2;
   end
   else
      l1 := l2;
   inicjuj(l2);
end; { polacz }

{ Scala dwie uporzadkowane rosnaco listy.
  W wyniku tworzy nowa liste. }
function scal(l1,l2 : lista) : lista;
var
   wsk1,wsk2,wynik,reszta : lista;
   dane1,dane2            : typ;
begin
   wsk1 := l1;
   wsk2 := l2;
   inicjuj(wynik);

   while (not pusta(wsk1)) and (not pusta(wsk2)) do
   begin
      dane1 := glowa(wsk1);
      dane2 := glowa(wsk2);
      if dane1 < dane2 then
      begin
         dodaj(wynik, dane1);
         wsk1 := ogon(wsk1);
      end else begin
         dodaj(wynik, dane2);
         wsk2 := ogon(wsk2);
      end
   end;

   if not pusta(wsk1) then
      reszta := wsk1
   else
      reszta := wsk2;

   while not pusta(reszta) do
   begin
      dodaj(wynik, glowa(reszta));
      reszta := ogon(reszta);
   end;

   scal := wynik;
end; { scal }

{ Sortuje liste rosnaco algorymem sortowania przez wstawianie. }
procedure sortuj(var l : lista);
var
   max, maxpop, akt, pop, wynik, reszta : lista;

   procedure znajdz_max;
   begin
      pop := nil;
      akt := reszta;

      maxpop := pop;
      max := akt;

      while not pusta(akt) do
      begin
         if glowa(akt) > glowa(max) then
         begin
            max := akt;
            maxpop := pop;
         end;
         pop := akt;
         akt := ogon(akt);
      end;
```

```
   end;

begin
   if pusta(l) then exit;

   reszta := l;
   inicjuj(wynik);

   while not pusta(reszta) do
      begin
         znajdz_max;

         if pusta(maxpop) then
            reszta := ogon(reszta)
         else
            maxpop^.ogon := ogon(max);

         max^.ogon := wynik;
         wynik := max;
      end;

   l := wynik;
end; { sortuj }

{ Usuwa liste z pamieci. }
procedure kasuj(var l : lista);
var
   ignoruj : typ;
begin
   while not pusta(l) do ignoruj := usun(l);
end; { kasuj }

{ Drukuje na ekranie cala liste. }
procedure drukuj(l : lista);
var
   wsk : lista;
begin
   wsk := l;
   if not pusta(wsk) then
   begin
      write(wsk^.dane);
      wsk := ogon(wsk);
   end;
   while not pusta(wsk) do
   begin
      write(' ', wsk^.dane);
      wsk := ogon(wsk);
   end;
end; { drukuj }

var
   l1,l2,l3 : lista;

begin
   inicjuj(l1);
   inicjuj(l2);

   dodaj(l1, 20);
   dodaj(l2, -90);
   dodaj(l1, 13);
   dodaj(l2, 27);
   dodaj(l1, 56);
   dodaj(l2, 1003);
   dodaj(l1, 46);
   dodaj(l2, -23);
   dodaj(l1, -6);
   dodaj(l2, 67);

   dodaj(l1, -75);
   dodaj(l2, 44);
   dodaj(l1, 6345);
   dodaj(l2, 313);
   dodaj(l1, 768);
   dodaj(l2, 2213);
   dodaj(l1, 868);
   dodaj(l2, -156);
   dodaj(l1, 9467);
   dodaj(l2, -230);

   write('l1 = |'); drukuj(l1); writeln('|');
```



```
   write('l2 = |'); drukuj(l2); writeln('|');

   sortuj(l1);
   write('posortowana l1 = |'); drukuj(l1); writeln('|');

   sortuj(l2);
   write('posortowana l2 = |'); drukuj(l2); writeln('|');

   l3 := scal(l1,l2);
   write('po scaleniu l3 = |'); drukuj(l3); writeln('|');

   polacz(l1,l2);
   write('po polaczeniu l1 = |'); drukuj(l1); writeln('|');
   write('po polaczeniu l2 = |'); drukuj(l2); writeln('|');

   kasuj(l1);
   kasuj(l2);

   write('po skasowaniu l1 = |'); drukuj(l1); writeln('|');
   write('po skasowaniu l2 = |'); drukuj(l2); writeln('|');
end.

{

Wynik dzialania programu na ekranie:

l1 = |9467 868 768 6345 -75 -6 46 56 13 20|
l2 = |-230 -156 2213 313 44 67 -23 1003 27 -90|
posortowana l1 = |-75 -6 13 20 46 56 768 868 6345 9467|
posortowana l2 = |-230 -156 -90 -23 27 44 67 313 1003 2213|
po scaleniu l3 = |9467 6345 2213 1003 868 768 313 67 56 46 44 27 20 13 -6 -23 -75 -90 -156 -230|
po polaczeniu l1 = |-75 -6 13 20 46 56 768 868 6345 9467 -230 -156 -90 -23 27 44 67 313 1003
                                                                                    2213|
po polaczeniu l2 = ||
po skasowaniu l1 = ||
po skasowaniu l2 = ||

}
```