```
unit Listy_P;

        interface

        type
        TDane  = byte;
     PLista = ^TLista;
     TLista = record
                                          dane             : TDane;
            wskaznik : PLista
                                    end;

   var
                          skladnik_biezacy, pierwszy_skladnik : PLista;
     element
: TDane;
     koniec
                                     : Boolean;
     k
                                      : integer;

   procedure EnList(var skladnik_biezacy:PLista; element:TDane);

   procedure AddressComponentList(var k:integer; pierwszy_skladnik:PLista;

                                                                                  var
skladnik_biezacy:PLista;
                                        var koniec:Boolean);

   procedure DeList(var pierwszy_skladnik, skladnik_biezacy:PLista;

                                                                        var
element:TDane);

   procedure Print(pierwszy_skladnik:PLista);

  implementation

                  procedure EnList(var skladnik_biezacy:PLista; element:TDane);
   var
        poprzedni_skladnik, nastepny_skladnik : PLista;
   begin
        if skladnik_biezacy <> nil then
        begin
        poprzedni_skladnik := skladnik_biezacy;
       nastepny_skladnik := skladnik_biezacy^.wskaznik
      end
     else
```

```
      begin
        poprzedni_skladnik := nil;
        nastepny_skladnik := nil
      end;
    New(skladnik_biezacy);
    with skladnik_biezacy^ do
      begin
         dane := element;
        wskaznik := nastepny_skladnik
      end;
    if poprzedni_skladnik <> nil then
                                  poprzedni_skladnik^.wskaznik := skladnik_biezacy
  end;

  procedure AddressComponentList(var k:integer; pierwszy_skladnik:PLista;

                                                                                  var
skladnik_biezacy:PLista;


                          var koniec:Boolean);
  var
    i : integer;
  begin
    if pierwszy_skladnik <> nil then
      if k = 1 then
        begin
          skladnik_biezacy := pierwszy_skladnik;
          if skladnik_biezacy^.wskaznik = nil then
                  koniec := TRUE
          else
            koniec := FALSE
        end
      else
                                        begin
          i := 1;
          repeat
            Inc(i);
            if pierwszy_skladnik^.wskaznik <> nil then
              pierwszy_skladnik := pierwszy_skladnik^.wskaznik
          until (pierwszy_skladnik^.wskaznik = nil) or (i = k);
          if pierwszy_skladnik^.wskaznik = nil then
                  koniec := TRUE
          else
            koniec := FALSE;
          if k > i then
            k := 0
          else
```

```
        skladnik_biezacy := pierwszy_skladnik
                                                end
  else
    begin
       k := -1;
     koniec := TRUE
    end
end;

procedure DeList(var pierwszy_skladnik, skladnik_biezacy:PLista;

                                                                    var element:TDane);
var
  poprzedni_skladnik, nastepny_skladnik : PLista;
begin
  if (pierwszy_skladnik <> nil) and (skladnik_biezacy <> nil) then
    if pierwszy_skladnik <> skladnik_biezacy then
      begin
        poprzedni_skladnik := pierwszy_skladnik;
        nastepny_skladnik := poprzedni_skladnik^.wskaznik;
        if nastepny_skladnik <> skladnik_biezacy then
                repeat
                poprzedni_skladnik := nastepny_skladnik;
          nastepny_skladnik := poprzedni_skladnik^.wskaznik
        until nastepny_skladnik = skladnik_biezacy;
        with skladnik_biezacy^ do
          begin
                element := dane;
          poprzedni_skladnik^.wskaznik := wskaznik
          end;
        Dispose(skladnik_biezacy);
        skladnik_biezacy := poprzedni_skladnik
      end
    else
      begin
        with pierwszy_skladnik^ do
          begin
                element := dane;
          pierwszy_skladnik := wskaznik
          end;
        Dispose(skladnik_biezacy);
        skladnik_biezacy := pierwszy_skladnik
      end
                end;

procedure Print(pierwszy_skladnik:PLista);
begin
  if pierwszy_skladnik = nil then
```

```
      WriteLn('Lista jest pusta')
    else
      repeat
        Write(pierwszy_skladnik^.dane, ' ');
        pierwszy_skladnik := pierwszy_skladnik^.wskaznik;
      until pierwszy_skladnik = nil
  end;
begin
  skladnik_biezacy := nil;
  pierwszy_skladnik := nil
end.
```