Bankkontoregister

Vi har en posttabell, REG, med fälten: kontonummer, namn, ranta och saldo och en funktion saldo = trans(accountno, amount). Låt oss bygga ut hanteringen av detta register. Man kan bl.a. tänka sig följande kompletteringar:

- Skapa ett konto
- Ta bort konto
- Redigera konto, t.ex. namnbyte eller rättelse av fel
- Hantera ränta, ev. med särskild hantering av konton med underskott

Låt oss börja med att ta bort ett konto. Funktionen kan ta kontonummer som inparameter, behöver inte returnera något, kan kommunicera med användaren och kan ha huvudet:

```
function killaccount(accountno)
```

Vi behöver göra det vi gjorde i `trans`, nämligen söka i registret efter en post med ett givet kontonummer. Då är det rimligt att göra funktionen

regfind anropbar från både `trans` och `killaccount`. Vi flyttar alltså funktionen till en fil med namnet `regfind.m`:

```
function index = regfind(accountno)
% Sök en post i REG med kontonummer accountno
% Returnera postens nummer
% eller -1 om sådan post inte finns
global REG
for no = 1:numel(REG)
  if REG(no).kontonummer==accountno
    index = no; return;
  end
end
index = -1;
```

I `killaccount` bör vi kontrollera indata, att det är en inparameter och att det är ett tal i tillåtet intervall. T.v. förutsätter vi att kontonummer ska ligga i intervallet 1..99 999.

```
function killaccount(accountno)
```

```
% Om accountno är ett tillåtet kontonummer och
% det finns ett konto med detta nummer tas posten
% med detta nummer bort ur REG
global REG
accountno = accountno(1);
if ~isnumeric(accountno)
    error('Kontonumret är ej ett tal.');
elseif mod(accountno,1)
    error('Kontonumret är ej ett heltal.');
elseif accountno<1||accountno>99999
    error(['Kontonumret ligger ej mellan 1 '...
    'och 99999.']);
end
recno = regfind(accountno);
if recno==-1
    error(['Det finns inget konto med nummer '...
    num2str(accountno)]);
```

```
else
    disp('Ska följande konto dödas?');
    fn = fieldnames(REG);
    for index = 1:numel(REG)
        disp([fn{index} ': ' ...
        getfield(REG(recno),fn{index})]);
    end
    answer = input('Svara ''J'' för ja: ','s');
    if upper(answer)~='J'
        disp('Kontot tas ej bort.');
    else
        REG = REG([1:recno-1 recno+1:end]);
        disp('Kontot är dödat.');
    end
end
```

Det skulle vara bra att kunna lagra även andra data än själva kontona i registret. Vi kan låta `REG` ha ett fält `maxaccount` som innehåller högsta tillåtna kontonummer och ett andra fält `account` som är kontoförteckningen.

Vi kan göra en kommandofil som initierar ett testregister:

```
% Initiera bankkontoregistret
global REG
REG.maxaccount = 99999;
REG.account.kontonummer = 12357;
REG.account.namn = 'Kloker';
REG.account.ranta = 2.2;
REG.account.saldo = 200;
REG.account(2).kontonummer = 314159;
REG.account(2).namn = 'Trötter';
REG.account(2).ranta = 1.9;
REG.account(2).saldo = 10500;
```

Vi måste ändra på några ställen i `trans` och `regfind`: ändra alla `REG` till `REG.account`
och ändra i `killaccount` till: `accountno>REG.maxaccount`, `' num2str(REG.maxaccount) '` och

```
fn = fieldnames(REG.account);
for index = 1:numel(fn)
  disp([fn{index} ': ' ...
  getfield(REG.account(recno),fn{index})]);
end
answer = input('Svara ''J'' för ja: ','s');
if upper(answer)~='J'
  disp('Kontot tas ej bort.');
else
  REG.account = REG.account([1:recno-1 ...
  recno+1:end]);
```

Registret är ju globalt så att det nås från alla funktioner där det är deklarerat globalt. Det försvinner dock när man avslutar Matlab. Vi kan spara registret på fil med `save REG REG` som sparar variabeln `REG` på filen `REG.mat` så att den kan hämtas med `load REG`.

Härefter kan den intresserade studenten göra andra tillägg till bankkontohanteringen.