Stor övningsuppgift inför laboration 3 – Textfilsbaserat register

Hela denna uppgift är endast en övning inför sista laborationen. Den här uppgiften ska inte redovisas med plagiatkontroll men det är ändå viktigt att ni behandlar den som om den skulle redovisas med plagiatkontroll. Den här uppgiften kommer att understödja sista laborationen som är uppgift A6. I denna stora övningsuppgift kommer ni att få arbeta med större program, ni ska utveckla förmågan att hantera ett större problem genom att dela upp det i småproblem. Det är vidare viktigt, när ni börjar med denna uppgift, att ni har läst och gärna övat på avsnittet 11.1 i boken och arbetat ordentligt med innehållet i föreläsning 11, om referensparametrar. Samt exemplet `matriserifunktioner.c` (Finns på KTH-Social under FL 9.)

Problemet som vi ska studera är det klassiska dataregistret. Vi ska skriva ett program liknande det i avsnitt 12.4 i läroboken och ni är fria att använda kod från det programmet. Vi ska dock göra en del förändringar i förutsättningarna. Vi ska skapa ett register med personuppgifter. Dessa personuppgifter ska lagras i en *textfil* och den textfilen ska ha utseendet

```
Johnny:johnnytheswede@yahoo.com:08-7909473
Nina:nina@ninashem.se:08-35345345
```

och så vidare, vi lagrar alltså namn, epostadress och telefonnummer i textformat och separerar dessa tre uppgifter med kolon. En rad i filen innehåller exakt en personuppgift. Programmet ska kunna hantera personuppgifter till högst 20 personer. Denna gräns kan förstås ökas enkelt, men den ska uttryckas i programmet genom makrot

```
#define MAX 20
```

och sedan ska programkoden referera till `MAX` överallt där det går och överallt där det behövs.

Programmet ska köra genom att först fråga om man ska arbeta med ett redan existerande register eller om man vill skapa ett nytt. Om man vill arbeta med ett redan existerande register ska programmet fråga efter filnamnet och öppna registret. Om man vill skapa ett nytt register ska programmet också fråga efter ett filnamn, men det här filnamnet avgör då under vilket namn det nya registret ska sparas. Efter denna inledning ska en meny komma upp som har 7 val:

1. Addera personuppgift till registret
2. Ta bort en personuppgift från registret
3. Ändra i personuppgift
4. Söka
5. Sortera
6. Skriva ut hela registret
7. Avsluta

Med dessa menyval kan man kontrollera hela registrets innehåll. Sökning ska kunna ske på *delar* av namn, epostadresser och telefonnummer, man ska alltså kunna mata in en del av ett namn eller en del av ett telefonnummer eller en del av en mejladress och få upp en lista på alla personuppgifter där det finns en överensstämmelse. Sortering ska kunna ske på alla tre sakerna, namn, epost eller telefonnummer men endast en i taget. Man får ange vilket som gäller. Utskrift av registret ska inte bara vara en utskrift av textfilens innehåll utan en utskrift av registrets nuvarande innehåll som ändras hela tiden när man arbetar med registret. Då vill vi även ha en bra formatering av de uppgifterna i stil med:

```
Namn: Johnny   Epost: johnnytheswede@yahoo.com    Tel: 08-7909473
```

och inte bara textraden

```
Johnny:johnnytheswede@yahoo.com:08-7909473.
```

Sökning och sortering ska vara okänsliga för stora och små bokstäver men registret ska innehålla personuppgifter med både stora och små bokstäver. (För att lösa detta, lös först problemet med sökning där sökningen *är* beroende av stora och små bokstäver, finn sedan ett sätt att *göra sökningen oberoende* av stora och små bokstäver.)

Funktionen ”`Ta bort`” måste föregås av en sökning: först söker man på vilken person man vill ta bort och sedan får man en fråga, vilken av de personer som uppfyller sökkriterierna vill du ta bort? Ett körexempel kan se ut så här:

```
Ta bort personuppgift:

Vad vill du söka på?

1. Namn
2. Epost
3. Telefonnummer

> 1

Sök efter person med namn: an

Personer med namn som överstämmer med sökkriteriet 'an' är

1. Anders:akarlsson@hemma.se:08-234234234
2. Anna:apettersson@yahoo.com:019-23423423

Vilken ska bort (0 för att ångra): 2

Tar bort 'Anna'
```

Vi ska här observera tre saker. För det första: sökningen är inte känslig för stora och små bokstäver. För det andra: det är endast en av personuppgifterna som tas bort och vilken anges av användaren. För det tredje: `Anders` och `Annas` personuppgifter kanske inte alls finns på plats 1 respektive 2 i registret, alltså i arrayerna som lagrar personuppgifterna, men de *kom på plats 1 respektive 2 i just den här sökningen*. Hade vi använt en annan söksträng, till exempel ann så hade kanske programmet ställt följande fråga:

```
Personer med namn som överstämmer med sökkriteriet 'ann' är

1. Anna:apettersson@yahoo.com:019-23423423

Vilken ska bort (0 för att ångra):
```

Det betyder alltså att programmet i samband med en sökning får ett specialiserat resultat bestående

av en specifik mängd personuppgifter. Denna specifika mängd kan beskrivas som en följd av indexnummer i den stora array som lagrar alla personuppgifter. Det är denna mängd av indexnummer som måste sparas och behandlas i en sökning. En speciell array kommer då att behövas till detta. Den kanske kan deklareras så här:

```
int search_result[MAX];
```

På samma sätt ska även en ändring av en personuppgift föregås av en sökning, vi letar efter den personuppgift som ska ändras och sedan väljer vi vilken som ska ändas och genomför ändringen. Det kan här vara bra att mata in tomma strängar om man vill behålla de gamla värdena i personuppgiften. Så här:

```
Ändra i personuppgift

Namn:  Johnny
Epost: johnnytheswede@yahoo.com
Tele:  08-7909473

Tryck bara return om uppgiften inte ska ändras

Nytt namn:
Ny epost:
Nytt telnr: 090-2342342

Ny personuppgift:

Namn:  Johnny
Epost: johnnytheswede@yahoo.com
Tele:  090-2342342
```

Då man skriver ut hela registret skrivs det ut i den ordning det för närvarande är sorterat i. Om man väljer ”`Sortera`” får man en fråga på vilken del man vill sortera (namn, epost eller telefonnumer) och då sorteras registret om. När man väljer ”`Avsluta`” så skrivs registrets innehåll in i den textfil vars filnamn bestämdes då programmet startade, antingen genom att man skapade ett nytt register eller genom att man valde att arbeta med ett existerande register.

I den här uppgiften är det viktigt att du inte skriver om samma kod flera gånger, du måste se till att det finns funktioner för de processer som förekommer flera gånger.

Ledning: arbeta med tre arrayer av strängar:

```
char name[MAX][20];
char email[MAX][30];
char phone[MAX][20];
```

Deklarera dessa som lokala variabler i main och skriv funktioner tar dessa tre arrayer i parametrar och förändrar innehållet. Senare, i sista laborationen, ska vi strukturera upp det här ordentligt och skapa en bättre hantering, men det väntar vi med till då.

Exempel: Funktionen `sortera` kan få följande (ungefärliga) funktionsprototyp

```
sortera (char name[][20], char email[][20], char phone[][20], int length);
```

där alltså referenser till hela registret som ligger i main skickas som referensparametrar till funktionen.