KTH – STH

TENTAMEN

`HI1024:TEN2` - *Praktisk tentamen*
Tid: 8-13, den 18 februari 2012

Gamla kurskoder: HI1900, 6E2950, etc.
Examinator: *Johnny Panrike*
Rättande lärare: *Nicklas Brandefelt*, *Johnny Panrike* och *Peter Steiner*
Tentamen konstruerad av de som rättar tentan.

Tentamen består av 5 sidor (inklusive denna sida.)

Instruktioner. Alla uppgifter är värda 4 poäng styck. Tentan har en **A**-del och en **B**-del. A-delen innehåller tre uppgifter och B-delen innehåller två uppgifter. För att få minsta godkända betyg, E, ska man klara 8 poäng på A-delen. Om man inte klarat A-delen rättas inte B-delen. Om man klarat A-delen har man minst betyget E. B-delens uppgifter kan då ge ett högre betyg.

Tillåtna hjälpmedel: Vad som helst i *pappersform*. Böcker, utskrifter, egna anteckningar etc. Dock inga som helst elektroniska hjälpmedel som USB-minnen, CD-skivor eller liknande.

- På `W:\` finns det tillhörande filer för tentamen
- Man behöver inte ta hänsyn till `ÅÄÖ` någon av uppgifterna.
- Svaren till uppgifterna ska lämnas i form av *C*-program **på nätverksenheten H:** med namnen `UPPG1.C`, `UPPG2.C`, `UPPG3.C`, `UPPG4.C`, `UPPG5.C`. Inga andra filer kommer att beaktas vid rättningen (förutom de filer som hör till olika resultat som ska produceras av de program ni skriver). På H: ska också finnas en textfil med ditt namn och personnummer i. Lägg dit den filen det första du gör så att vi vet att du har kontakt med H:.

För att ett program ska rättas måste det kunna köra (och därmed definitivt kompileras.) Rättningen går till som så att programmet provkörs för några olika indata, ofta andra än de som ges på själva tentamen. Programmet ska ge korrekt resultat vid dessa körningar. Vidare ska även programmets form och innehåll uppfylla de krav som ställts i uppgiften, helt enkelt till exempel om man ska skriva en funktion så ska en funktion finnas i svaret som man ger in.

Alla uppgifter ger 4 poäng. Man kan få delpoäng för halvt fungerande program, men man får aldrig poäng för program som inte ens kan kompilera och köra. På samma sätt ges inga poäng för program som kraschar. (Till exempel om man glömmer adressoperatorn, `&`, då den behövs till `scanf()`.)

Uppgifterna i Del A examinerar också det som benämnts som *kärnkompetensen* i programmering. Vid varje uppgift anges vilket delområde av kärnkompetensen som de examinerar.

Del A

1 Jämför strängar (4p) *[Strukturering, format/protokoll]*

Skriv en funktion som tar två strängar och returnerar hur många av bokstäverna i den första strängen som också finns i den andra strängen.

Ex:
`comp("banan","arbete")` ska returnera `3` eftersom `b`,`a` och `a` i banan finns i arbete
`comp("banan","avlusa")` ska returnera `2` eftersom `a` och `a` i banan finns i avlusa
`comp("rivas","avlusa")` ska returnera `3` eftersom `v`,`a` och `s` i rivas finns i avlusa

Skriv också en `main()` som testar funktionen.

Exempel på `main()` som testar funktionen:

```
int main(void)
{
    printf("Antal: %d",comp("banan","arbete"));
}
```

2 Ord som också är ord baklänges (4p) *[Strukturering, funktioner]*

Skriv ett program som låter användaren mata in ett ord och som sedan berättar om ordet blir ett ord baklänges. Till din hjälp har du filen `SAOL12.txt` där det finns över 90000 ord. En testkörning av ett sådant program kan se ut så här:

```
Vilket ord vill du kontrollera? (skriv med stora bokstaver)PUTS
Ordet blir ett ord baklanges namligen: STUP
```

Du kan utgå från att användaren skriver med stora bokstäver och att ordet hon skriver faktiskt är ett ord. Viktigt: programmet **ska** bygga på en funktion som vänder på ett ord i en sträng och använda den funktionen på strategiska ställen i programmet.

3. Kurs som struct (4p) *[Strukturering]*

En kurs på KTH har flera egenskaper, en kod, ett namn, moment i LADOK och förkunskaper. Till exempel har den här kursen kurskoden HI1024, namnet ”Programmering, grundkurs”, momenten TEN1, TEN2 och LAB1. Kursen kräver (formellt sett) inga förkunskaper, men vi skulle kunna kräva att kursen HF1010, som heter ”Introduktion till datateknik” är förkunskaper.
Vi representerar en kurs så här:

```
struct kurs
{
  char kod[7];
  char namn[50];
  char moment[20];
  char forkunskaper[7];
};
```

En binärfil är baserad på ovanstående struct och innehåller information om en del kurser som ges i ett fiktivt utbildningsprogram som handlar om... ja det får ni se när ni öppnar filen! Skriv ett program som öppnar denna fil och läser in innehållet till en array av structar och sedan skriver ut kurserna på följande sätt:

```
Kurser:
1. namn
2. namn
3. namn
...
```

Därefter ska programmet fråga efter en kurs som man får mata in, fullständig information om kursen visas då samt alla den kursens förkunskaper. Exempel

```
Kurs: JS1234, Flugfiske
Moment: TEN1 LAB1

Förkunskaper:
JS1230, Fiske
```

I den här situationen är Flugfiske en kurs som har Fiske som förkunskaper. (Vi bortser från fallet om en kurs har en kurs som förkunskaper som i sin tur har ytterligare en kurs som förkunskaper.)

Del B

4. Glosor (4p)

Skriv ett program som förhör engelska glosor. Programmet ska ge möjlighet att spara och läsa upp glosor från fil (du väljer om programmet lagrar ord i textfil eller binärfil).

Programmet startar med att fråga om användaren vill läsa in glosor från fil. Om ja får användaren skriva in filnamn och sedan läses glosorna in.

Sedan får användaren upp en meny:

```
1. Skriva in nya glosor
2. Ova svenska till engelska
3. Avsluta
```

1. Skriva in nya glosor.
Användaren får skriva in ord på engelska och svenska tills hon slår enter eller skriver stopp eller annan lösning.

2. Öva svenska till engelska
Användaren får välja antal glosor hon vill öva på. Programmet presenterar sedan slumpmässigt ett ord på svenska och användaren får sedan skriva in det engelska. När rätt antal ord är övade får användaren veta hur många procent rätt hon hade. Samma ord kan övas flera gånger.

3. Avsluta
Vid avsluta får användaren ange vilken fil hon vill spara till. Väljer hon en fil som redan finns skrivs denna över. Sedan avslutar programmet.

Om du vill kan du utgå från följande skelett när du programmerar:

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <time.h>

#define maxOrdlangd 30
#define maxAntalOrd 100

int main(void)
{
    char lasa[5];
    int antalOrd=0;
    int menyval;
    char svenska[maxAntalOrd][maxOrdlangd];
    char engelska[maxAntalOrd][maxOrdlangd];

    srand(time(0));

    printf("Vill du lasa in glosor fran fil (ja/nej)?");
    scanf("%s",lasa);
    fflush(stdin);
    if(strcmp("ja",lasa)==0)
    {
            lasIn(svenska,engelska,&antalOrd);
    }
    do
    {
        printf("1. Skriva in nya glosor\n");
        printf("2. Ova svenska till engelska\n");
        printf("3. Avsluta\n");
        scanf("%d",&menyval);
        if(menyval==1)
        {
            skrivIn(svenska,engelska,&antalOrd);
        }
        else if(menyval==2)
        {
            ova(svenska,engelska,antalOrd);
        }

    }while(menyval!=3);
    skrivTillFil(svenska,engelska,antalOrd);

}
```

Kvar att göra då är att skriva funktionerna.

5. Master Mind (4p)

I det här spelet slumpar datorn fram 4 siffror (0-9) på fyra positioner. Spelaren skall sedan gissa vilka siffror datorn har slumpat fram. Efter gissningen berättar datorn hur många siffror som var rätt siffra på rätt plats och hur många siffror som var rätt siffra men på fel plats. Sedan får spelaren gissa igen och så fortsätter spelet tills spelaren gissat rätt varvid denne gratuleras av programmet. Datorn slumpar fram fyra unika siffror och spelaren förväntas gissa fyra unika siffror.

Så här skulle en körning där datorn slumpat fram siffrorna 3 7 1 0 se ut:

```
Valkommen. Gissa fyra siffror(0-9) med mellanslag mellan
siffrorna:
0 2 6 3
Du har 0 siffror ratt pa ratt plats och 2 siffror ratt pa fel
plats. Gissa igen:
2 7 4 0
Du har 2 siffror ratt pa ratt plats och 0 siffror ratt pa fel
plats. Gissa igen:
2 3 1 0
Du har 2 siffror ratt pa ratt plats och 1 siffror ratt pa fel
plats. Gissa igen:
3 7 1 0
Grattis du gissade ratt på 4 gissningar!
```