Örebro universitet
Akademin för naturvetenskap och teknik
Thomas Padron-McCarthy (thomas.padron-mccarthy@oru.se)

Tentamen i

Programmering grundkurs och Programmering C

för D1 m fl, även distanskursen

fredag 14 januari 2011

Gäller som tentamen för:
DT1016 Datateknik A, Programmering grundkurs, provkod 0100
DT1007 Datateknik A, Tillämpad datavetenskap, provkod 0410
DT1006 Datateknik A, Programmering C, distans, provkod 0100

Hjälpmedel:	Inga hjälpmedel.
Poängkrav:	Maximal poäng är 37. För godkänt betyg (3 respektive G) krävs 20 poäng. För den som följt campuskursen hösten 2010 ger varje i tid inlämnad inlämningsuppgift med deadline en extra poäng. Den som *inte* gått campuskursen hösten 2010 får dessa (tre) extrapoäng ändå.
Resultat och lösningar:	Meddelas via e-post eller på kursens hemsida, http://basen.oru.se/kurser/c/2010-2011-p2/, senast fredag 4 februari 2011.
Återlämning av tentor:	Efter att resultatet meddelats kan tentorna hämtas på universitetets centrala tentamensutlämning.
Examinator och jourhavande:	Thomas Padron-McCarthy, telefon 070-73 47 013.

- Skriv tydligt och klart. Lösningar som inte går att läsa kan naturligtvis inte ge några poäng. Oklara och tvetydiga formuleringar kommer att misstolkas.
- Skriv den personliga tentamenskoden på varje inlämnat blad. Skriv *inte* namn eller personnummer på bladen.
- Skriv bara på en sida av papperet. Använd inte röd skrift.
- Antaganden utöver de som står i uppgifterna måste anges.
- Skriv gärna förklaringar om hur du tänkt. Även ett svar som är fel kan ge poäng, om det finns med en förklaring som visar att huvudtankarna var rätt.

LYCKA TILL!

Prioritet och associativitet hos operatorerna i C

De viktigaste operatorerna:

Prioritet	Kategori	Operator	Associativitet
Högsta	Unära postfixoperatorer	(), [], ->, ., ++, --	vänster
	Unära prefixoperatorer	!, ++, --, +, -, *, &, sizeof, (*typ*)	höger
	Multiplikation mm	*, /, %	vänster
	Addition mm	+, -	vänster
	Jämförelser	<, <=, >=, >	vänster
	Likhetsjämförelser	==, !=	vänster
	Logiskt OCH	&&	vänster
	Logiskt ELLER	||	vänster
Lägsta	Tilldelning	=, +=, -=, *=, /=, %=	höger

Uppgift 1 (1 p)

Vilka värden har följande uttryck?

a) 1 * 2 + 3 * 4

b) 1 + 2 * 3 + 4

c) 1 / 2 + 3 % 4

Uppgift 2 (1 p)

Variablerna **a**, **b**, **c** och **d** är av typen **int**. Vilka värden har variablerna efter att följande kod har körts?

```
a = 1; b = a + 2; c = 3; d = 4;
while (a < b) {
    if (b == c && b == d)
        a = a + 2;
    else
        a = a + 3;
}
```

Uppgift 3 (3 p)

Skriv ett komplett C-program (med **#include** och allt) som läser in tre heltal, vi kan kalla dem **a**, **b** och **c**, och sen skriver ut alla tal från **a** till **b** (inklusive de talen) utom talet **c** (om det nu ligger mellan **a** och **b**).

I den här och alla andra uppgifter på tentan gäller: Normalt är felhantering en stor del av ett program. Vad ska till exempel hända om användaren skriver **Kalle** när hon egentligen borde mata in ett tal? Här behövs dock ingen felhantering, om så inte särskilt efterfrågas i uppgiften.

Uppgift 4 (2 p)

Skriv en funktion **korrekt_ordning**, som tar tre flyttal som argument och returnerar talet **1** om de är ordnade i storleksordning så att det första flyttalet är mindre än det andra, och det andra är mindre än det tredje. Annars ska funktionen returnera talet **0**.

Uppgift 5 (2 p)

Skriv en **main**-funktion som först läser in tre flyttal från användaren, sedan anropar funktionen **korrekt_ordning** för att avgöra om de är ordnade i storleksordning, och till sist skriver ut ett meddelande om ifall de är ordnade eller inte.

I den här och alla andra uppgifter på tentan gäller: Om du behöver använda något från en tidigare uppgift eller deluppgift, till exempel anropa en funktion som skrevs i den tidigare uppgiften, så behöver du inte skriva samma kod igen. Du får också göra uppgiften även om du inte gjort den tidigare uppgiften.

Uppgift 6 (2 p)

Svenska Postpropplotteriet är en ondskefull organisation som har som mål att proppa igen folks brevlådor med reklam, så att det bildas så kallade postproppar.

Egentligen ville Svenska Postpropplotteriet ha en databas för att lagra data om postpropparna, men av etiska skäl vägrar Oracle, Microsoft och de andra tillverkarna av databashanterare att hjälpa dem. Därför ska de i stället göra ett C-program som hanterar postproppar.

Först ska vi skapa datatypen **struct Postpropp**, som används för att representera en postpropp. En postpropp har en **volym**, som mäts i kubikdecimeter och lagras som ett flyttal, och den har också ett **antal**, som är ett heltal och som anger hur många reklamförsändelser som postproppen består av. Dessutom ska man lagra **adressen** till brevlådan med postproppen. Adressen kan innehålla högst 60 tecken.

Vi ska också använda **#define** för att skapa ett makro för max-adresslängden.

Skapa posttypen och makrot, i rätt ordning så att det går att kompilera.

Uppgift 7 (1 p)

Definiera en variabel av typen **struct Postpropp** och initiera den med data om en postpropp som är 14,7 kubikdecimeter stor, innehåller 198 reklamförsändelser, och finns på **Granvägen 6A, 702 21 Örebro**.

Uppgift 8 (2 p)

Vi vill kunna visa postproppsposternas innehåll på skärmen. Skriv en funktion som heter **visa_postpropp**, och som skriver ut data om en postpropp. Funktionen ska ta en postproppspost (eller, om du vill, en pekare till den) som parameter. Här är ett exempel på hur utskriften kan se ut:

```
Volym: 14.7 kubikdecimeter
Antal reklamförsändelser: 198
Adress: Granvägen 6A, 702 21 Örebro
```

Uppgift 9 (3 p)

Skriv en funktion som heter **las_postpropp**, och som läser in data om en postpropp. Funktionen ska skriva ut lämpliga ledtexter på standardutmatningen, och läsa in data från standardinmatningen (som normalt är kopplad till tangentbordet).

Du får själv välja om du vill att funktionshuvudet ska se ut så här:

```
struct Postpropp las_postpropp()
```

eller så här:

```
void las_postpropp(struct Postpropp *p)
```

Uppgift 10 (1 p)

Skriv en funktion som heter **mindre_postpropp**, och som tar två postproppar som argument. Den ska returnera ett sant värde om den första postproppen har mindre volym än den andra, och ett falskt värde om den är lika stor eller större.

Uppgift 11 (2 p)

Skriv en main-funktion som har två lokala variabler av typen **struct Postpropp**, och som läser in data om två postproppar till dessa variabler med hjälp av funktionen **las_postpropp**. Därefter ska programmet skriva ut den största av de två postpropparna med hjälp av funktionen **visa_postpropp**. Använd funktionen **mindre_postpropp** för att avgöra vilken som är störst.

Uppgift 12 (3 p)

Skriv funktionen **biggest_propp**, som tar en array av postproppar som argument, tillsammans med antalet postproppar i arrayen, och som returnerar den största av postpropparna i arrayen. (Använd gärna funktionen **mindre_postpropp** när funktionen söker efter den största proppen, men det är inget krav.)

Uppgift 13 (5 p)

Skriv ett C-program som läser in postproppar med hjälp av funktionen **las_postpropp**, och sparar allihop i en array. Arrayens ska kunna innehålla högst 2000 postproppar, och dess storlek ska anges med ett makro. Välj själv hur inmatningen ska avslutas. (Det ska inte vara ett förutbestämt antal postproppar.)

Därefter ska programmet anropa funktionen **biggest_propp** för att hitta den största proppen, och avslutningsvis skriva ut den med funktionen **visa_postpropp**.

Uppgift 14 (4 p)

a) Skriv en funktion **spara_proppar** som sparar posterna i arrayen från uppgiften ovan på en fil. Välj själv om det ska vara en textfil eller en binärfil.

b) Visa hur funktionen ska anropas från programmet i uppgiften ovan, och eventuell övrig kod som behöver läggas till i det programmet.

Uppgift 15 (5 p)

Små postproppar är för lätta att få bort ur brevlådorna, så Svenska Postpropplotteriet bryr sig inte om dem. Skriv därför ett C-program som först frågar efter en volymgräns, läser filen från uppgiften ovan, och skriver en ny fil med samma format men ett nytt namn, som innehåller de postproppar från originalfilen som är större än volymgränsen.
