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 13 januari 2012

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 40. För godkänt betyg (3 respektive G) krävs 22 poäng. För den som följt campuskursen hösten 2011 ger varje i tid inlämnad inlämningsuppgift med deadline en extra poäng. Den som *inte* gått campuskursen hösten 2011 får dessa (tre) extrapoäng ändå.
Resultat och lösningar:	Meddelas via e-post eller på kursens hemsida, http://basen.oru.se/kurser/c/2011-2012-p2/, senast fredag 3 februari 2012.
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 C-uttryck?

a) 1 + 2 * 3 + 4

b) 1+2 * 3+4

c) 1 < 2 && 3 > 4

d) 1 < 2 || 3 / 0 > 4

Uppgift 2 (1 p)

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

```
a = 1; b = 2; x = 3.0; y = 4.0;
while (a < b) {
    if (x > y)
        a = a + 1;
    else
        x = x + 0.6;
    y = y + 0.1;
}
```

Uppgift 3 (3 p)

Skriv ett komplett C-program (med **#include** och allt) som läser in två flyttal. Om de är lika stora ska programmet bara skriva **Lika**, annars ska det skriva ut talen i storleksordning, med det minsta först.

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 (1 p)

Skriv en funktion, **korrekt_summa**, som tar tre heltal (vi kan kalla dem **a**, **b** och **c**) som argument, och som returnerar talet **1** om summan av **a** och **b** är lika med **c**. Annars ska funktionen returnera talet **0**.

Uppgift 5 (3 p)

Ett program ska hjälpa oss att öva på huvudräkning. Skriv en **main**-funktion som slumpar fram två heltal, skriver ut dem, och frågar efter deras summa. Om användaren matar in rätt summa ska programmet skriva **Rätt**, annars **Fel**. Programmet ska använda funktionen **korrekt_summa** för att avgöra om summan var rätt.

Tips: Funktionen **rand** ger ett slumpmässigt heltal.

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 (1 p)

En "app", som är förkortning för "applikation", är ett litet (eller ibland inte så litet) program för mobiltelefoner. Vi tänker sälja appar i vår on-line-affär, och behöver därför skapa datatypen **struct App**, som används för att representera en app. En app har ett **namn** som kan innehålla högst 80 tecken, ett **pris** som anges med ett flyttal, och ett **antal**, som är hur många gånger appen har laddats ner.

Vi ska också använda **#define** för att skapa ett makro för maxnamnlängden.

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

Uppgift 7 (1 p)

Definiera en variabel av typen **struct App** och initiera den med data om en app som heter **Satellite Rush**, som är gratis, och som laddats ner **500** gånger.

Uppgift 8 (2 p)

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

```
Namn: Satellite Rush
Pris: 0.00
Antal nerladdningar: 500
```

Uppgift 9 (3 p)

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

Du får själv bestämma hur den inlästa posten ska returneras.

Uppgift 10 (2 p)

Skriv en funktion **appswap** som byter plats på innehållet i två app-variabler.

Uppgift 11 (2 p)

Skriv en **main**-funktion som har två lokala variabler av typen **struct App**, och som läser in data om två appar till dessa variabler med hjälp av funktionen **las_app**. Därefter ska programmet först byta plats på innehållet i de två app-variablerna med hjälp av funktionen **appswap**, och sedan skriva ut den dyraste av de två apparna med hjälp av funktionen **visa_app**.

Uppgift 12 (5 p)

Vi vill spara apparna på textfilen **appar.txt**, där varje app lagras på tre rader (namn, pris och antalet nerladdningar), så här:

```
Satellite Rush
0.00
500
```

Skriv ett program som upprepat läser in appar med **las_app**, ända tills man matar in en app med namnet **END OF APPS**, och sparar alla apparna (utom den där sista) på textfilen.

Om filen inte går att öppna för skrivning, ska ett informativt och rättvisande meddelande skrivas ut, och programmet ska avslutas.

Uppgift 13 (6 p)

Skriv ett program som läser textfilen från uppgiften ovan, och skriver ut totalsumman som vi sålt appar för.

Om filen inte går att öppna för läsning, ska ett informativt och rättvisande meddelande skrivas ut, och programmet ska avslutas.

Uppgift 14 (4 p)

När man styrketränar kan man lyfta så tunga vikter som man klarar, så många gånger som man orkar. Men det kan ge bättre resultat att följa ett mer avancerat schema, där man lyfter olika vikter olika antal gånger. Ett exempel (som inte nödvändigtvis är bra) skulle kunna vara ett "pyramidschema" där man först bestämmer hur mycket man maximalt orkar lyfta, ens så kallade "one rep max". Sen lyfter man 10 procent av den vikten 10 gånger, 20 procent 9 gånger, 30 procent 8 gånger, och så vidare, ända tills 100 procent av vikten en gång.

Här gjorde vi tio så kallade "set" med olika vikter. Man kan också tänka sig till exempel tjugo sådana set, och då börjar man med 5 procent av vikten som man lyfter 20 gånger, därefter 10 procent 19 gånger, och så vidare, ända tills 100 procent av vikten en gång.

Skriv ett program där man matar in maxvikten och antalet set, och som sedan skriver ut träningsschemat. Avrunda vikterna till närmaste heltal.

Exempel:

Om användaren matar in maxvikten **100 kilo** och antalet **4**, ska träningsschemat bli **25 kg x 4, 50 kg x 3, 75 kg x 2, 100 kg x 1**.
Om Hulken matar in maxvikten **1000 kilo** och antalet **3**, ska träningsschemat bli **333 kg x 3, 667 kg x 2, 1000 kg x 1**.

Uppgift 15 (5 p)

Skriv ett program som läser in naturliga tal (alltså heltal som är större än eller lika med noll) från standardinmatningen. Inmatningen avslutas med ett negativt tal. Högst hundra tal kan matas in. Efter avslutad inmatning ska programmet tala om vilket tal som förekom flest gånger.

Om inget enskilt tal var vanligast, utan flera olika tal förekom lika många gånger, spelar det inget roll vilket tal som anges.
