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

Tentamen i

Programmering i språket C

för D1 m fl, även distanskursen

onsdag 30 maj 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 36. För godkänt betyg (3 respektive G) krävs 18 poäng.
Resultat och lösningar:	Meddelas via e-post eller på kursens hemsida, http://basen.oru.se/kurser/c/2011-2012-p2/, senast onsdag 20 juni 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 * 1 + 1 * 1

b) 1 + 1 * 1 + 1

c) 1+1 * 1+1

d) 1 - 1 - 1 - 1

Uppgift 2 (1 p)

Variablerna **ali**, **bodil** och **charlie** är av typen **int**. Vilka värden har variablerna efter att följande kod har körts?

```
ali = 1;
bodil = 2;
if (ali > bodil) {
    charlie = ali + bodil;
}
else {
    while (ali < bodil) {
        if (ali < 3)
            ++bodil;
        ++ali;
    }
    charlie = ali - bodil;
}
```

Uppgift 3 (3 p)

Skriv ett komplett C-program (med **#include** och allt) som låter användaren skriva in 357 heltal. När de 357 heltalen är inmatade, ska programmet skriva ut **Ok** om alla heltalen var mindre än 45. Annars ska programmet skriva ut **För stort**.

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, **inom_intervallet**, som tar tre flyttal som argument: ett tal, vi kan kalla det **x**, samt en nedre och en övre gräns. Om talet **x** ligger inom intervallet som anges av den nedre och övre gränsen, ska funktionen returnera ett sant värde. Annars ska funktionen returnera ett falskt värde. Tal som ligger precis på en gränserna räknas som inom intervallet.

Uppgift 5 (1 p)

Vi vill provköra funktionen **inom_intervallet**. Skriv därför en **main**-funktion som läser in tre tal, anropar **inom_intervallet** med de talen som argument, och skriver ut resultatet.

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)

Vi ska jobba mer med intervall, som i uppgifterna ovan. Skapa därför posttypen **Intervall**, som innehåller intervallgränserna. De är flyttal.

Uppgift 7 (2 p)

Skriv en funktion som heter **inside_interval**, och som tar två argument: ett flyttal och ett intervall av typen **Interval** (eller, om du vill, en pekare till det). Om det angivna flyttalet ligger inom intervallet, ska funktionen returnera ett sant värde. Annars ska funktionen returnera ett falskt värde.

Använd den tidigare funktionen **inom_intervallet**, som ska anropas från **inside_interval**.

Uppgift 8 (3 p)

Skriv en funktion som heter **overlapping_intervals**, och som tar två intervall (eller, om du vill, pekare) som argument. Om intervallen överlappar (dvs, det finns minst en punkt på tallinjen som tillhör båda intervallen), ska funktionen returnera ett sant värde. Annars ska funktionen returnera ett falskt värde.

Uppgift 9 (3 p)

Skriv en funktion som heter **inside_how_many_intervals**. Den ska gå igenom en array av intervall, och returnera hur många av dessa som ett angivet tal ligger inom. Funktionen ska ta tre argument: ett flyttal, en array av intervall, och ett heltal som anger antalet intervall i den arrayen.

Använd den tidigare funktionen **inside_interval**, som ska anropas från **inside_how_many_intervals**.

Uppgift 10 (3 p)

Vi vill provköra funktionen **inside_how_many_intervals**. Skriv därför en **main**-funktion som anropar funktionen med lämpliga data, och kontrollerar att den ger det förväntade svaret. (För en bra testning på riktigt skulle man behöva fler anrop, men här nöjer vi oss med ett enda.) Om funktionen ger fel svar, ska ett tydligt felmeddelande skrivas ut. Annars behöver inget skrivas ut.

Uppgift 11 (5 p)

Textfilen **intervall.txt** innehåller ett antal intervall, specificerade som lägsta och högsta gräns med två tal på varje rad. Skriv ett C-program som läser den filen, och skriver ut intervallgränserna för de två intervall som är smalast respektive bredast. Om filen inte går att öppna för läsning, ska ett felmeddelande skrivas ut, och programmet ska avslutas.

Uppgift 12 (6 p)

Vi vill kontrollera om filen med intervall i uppgiften ovan innehåller några överlappande intervall. Skriv ett C-program som gör det, och antingen skriver ut **Överlappande finns** eller **Överlappande finns INTE**. Om filen inte går att öppna för läsning, ska ett felmeddelande skrivas ut, och programmet ska avslutas.

Tips:

- Man måste jämföra varje intervall med vart och ett av alla de andra intervallen, och kontrollera om dessa två intervall överlappar.
- Använd funktionen **overlapping_intervals**.

Uppgift 13 (5 p)

Vi vill kontrollera amerikanska namn.

Amerikaner brukar skriva sina namn med ett förnamn, en initial med en punkt, och ett efternamn, till exempel **George W. Bush**, **Richard M. Nixon** och **John F. Kennedy**. Andra tillåtna namn är **Foo B. Fum**, **Zwz X. Wnnnn** och **A B. C**. Exempel på namn som *inte* är tillåtna är **Bill Clinton** (saknar initial) och **a b. c** (alla delarna måste börja med stor bokstav).

Skriv ett C-program som upprepat läser in ett namn på en rad, och talar om ifall det är ett tillåtet amerikanskt namn eller inte. Programmet ska avslutas så fort användaren matar in en tom rad.
