Örebro universitet
Institutionen 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

lördag 19 januari 2013

Gäller som tentamen för:
DT1029 Datateknik A, Programmering grundkurs, provkod 0100
DT1030 Datateknik A, Tillämpad datavetenskap, provkod 0410
DT1006 Datateknik A, Programmering C, distans, provkod 0100
DT1016 Datateknik A, Programmering grundkurs, provkod 0100
DT1007 Datateknik A, Tillämpad datavetenskap, provkod 0410

Hjälpmedel:	Inga hjälpmedel.
Poängkrav:	Maximal poäng är 37. För godkänt betyg (3 respektive G) krävs 21 poäng. För den som följt campuskursen hösten 2012 ger varje i tid inlämnad inlämningsuppgift med deadline en extra poäng. Den som *inte* gått campuskursen hösten 2012 får dessa (tre) extrapoäng ändå.
Resultat och lösningar:	Meddelas via e-post eller på kursens hemsida, http://basen.oru.se/kurser/c/2012-2013-p2/, senast lördag 9 februari 2013.
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

Några användbara biblioteksfunktioner

stdlib.h

```
int rand(void);
void srand(unsigned int seed);
void *malloc(size_t size);
void *realloc(void *ptr, size_t size);
void free(void *ptr);
void exit(int status);
void qsort(void *base, size_t nmemb, size_t size,
           int(*compar)(const void *, const void *));
```

stdio.h

```
FILE *fopen(const char *path, const char *mode);
int fclose(FILE *stream);
int getc(FILE *stream);
int getchar(void);
int ungetc(int c, FILE *stream);
char *fgets(char *s, int size, FILE *stream);
char *gets(char *s);
int putc(int c, FILE *stream);
int printf(const char *format, ...);
int fprintf(FILE *stream, const char *format, ...);
int sprintf(char *str, const char *format, ...);
int snprintf(char *str, size_t size, const char *format, ...);
int scanf(const char *format, ...);
int fscanf(FILE *stream, const char *format, ...);
int sscanf(const char *str, const char *format, ...);
size_t fread(void *ptr, size_t size, size_t nmemb, FILE *stream);
size_t fwrite(const void *ptr, size_t size, size_t nmemb, FILE *stream);
```

string.h

```
size_t strlen(const char *s);
char *strcpy(char *dest, const char *src);
char *strncpy(char *dest, const char *src, size_t n);
int strcmp(const char *s1, const char *s2);
int strncmp(const char *s1, const char *s2, size_t n);
char *strcat(char *dest, const char *src);
char *strncat(char *dest, const char *src, size_t n);
char *strstr(const char *haystack, const char *needle);
void *memmove(void *dest, const void *src, size_t n);
```

ctype.h

```
int isalnum(int c);
int isalpha(int c);
int isblank(int c);
int isdigit(int c);
int islower(int c);
int isprint(int c);
int ispunct(int c);
int isspace(int c);
int isupper(int c);
```

Uppgift 1 (1 p)

Vilka värden har följande C-uttryck?

a) 2*2 + 2/2

b) 2+2 * 2+2

c) 2 - 2 - 2 - 2

d) (2 + 2 / 2) / 2

Uppgift 2 (1 p)

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

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

Uppgift 3 (4 p)

Skriv ett komplett C-program (med **#include** och allt) som först frågar efter ett heltal. Därefter ska programmet läsa in så många flyttal som angavs av heltalet, och till sist skriva ut ur många av de flyttalen som var negativa.

> 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 (4 p)

Skriv ett komplett C-program (med **#include** och allt) som först läser in två rader (med högst 100 tecken på varje), och sedan slumpmässigt väljer en av de två raderna och skriver ut den.

> I den här och alla andra uppgifter på tentan gäller:
> Normalt ska man aldrig använda funktionen **gets**, utan i stället till exempel **fgets**. Här kan du dock använda **gets**.

Scenario

Vi ska skapa en webbplats som heter **Prisvakt**, och som hjälper användarna att hålla koll på priser på varor i olika butiker. Till exempel vill vi hålla reda på att en **Apple MacBook Air** kostar **8819** kronor på **CDON**, att samma dator kostar **9490** kronor på **Elgiganten**, och att en **Samsung 840 Series MZ-7TD500 500GB** kostar **2790** kronor på **Dustin Home**.

Till hjälp har vi en fil med varor och priser som heter **allapriser.txt**. Den innehåller prisuppgifter om varor. Varje prisuppgift lagras på tre rader: först namnet på varan, sedan namnet på butiken, och sist priset på den varan i den butiken. Så här ser filen ut:

```
Apple MacBook Air
CDON
8819
Apple MacBook Air
Elgiganten
9490
Samsung 840 Series MZ-7TD500 500GB
Dustin Home
2790
Samsung 840 Series MZ-7TD500 500GB
NetOnNet
2789
Apple MacBook Air
Dustin Home
9590

(och så vidare, med många miljoner prisuppgifter)
```

Uppgift 5 (2 p)

Skapa datatypen **struct Prisuppgift**, som innehåller en prisuppgift. En prisuppgift består av namnet på en vara, namnet på en butik, och priset på den varan i den butiken. Priset är ett heltal och namnen består av text med högst 40 tecken.

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 6 (1 p)

Definiera en variabel av typen **struct Prisuppgift** och initiera den med data om att en **Apple MacBook Air** kostar **9590** kronor på **Dustin Home**.

I den här och alla andra uppgifter på tentan gäller:
Om du behöver använda något från en tidigare uppgift eller deluppgift, till exempel utnyttja en datatyp eller 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 7 (1 p)

Skriv funktionen **samma_vara**. Funktionen ska ta två prisuppgifter som argument, och om de gäller samma vara, ska funktionen returnera ett sant värde. Annars ska den returnera ett falskt värde.

Uppgift 8 (2 p)

Skriv funktionen **billigare**, som jämför två prisuppgifter. Funktionen ska ta två prisuppgifter som argument, och om de gäller samma vara och det första av de två priserna är lägre än det andra, ska funktionen returnera ett sant värde. Annars ska den returnera ett falskt värde. Funktionen **billigare** ska använda sig av funktionen **samma_vara**.

Uppgift 9 (2 p)

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

```
Vara: Apple MacBook Air
Butik: Dustin Home
Pris: 9590
```

Uppgift 10 (3 p)

Skriv en funktion som heter **las_prisuppgift**, och som läser in en prisuppgift. 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 11 (3 p)

Skriv en **main**-funktion som har två lokala variabler av typen **struct Prisuppgift**, och som läser in två prisuppgifter till dessa variabler med hjälp av funktionen **las_prisuppgift**. Därefter ska den anropa funktionen **samma_vara**, och tala om för användaren om de två prisuppgifterna handlade om samma vara eller inte. Om de handlade om samma vara, ska den skriva ut den billigaste med hjälp av funktionen **visa_prisuppgift**. För att avgöra vilken som är billigast ska den använda funktionen **billigare**.

Uppgift 12 (8 p)

Som det står i scenariot finns alla prisuppgifterna på textfilen **allapriser.txt**.

Skriv ett program som frågar efter namnet på en vara, och sedan talar om i vilken butik den är billigast, och vad den kostar där.

Om filen inte går att öppna för skrivning, eller om den sökta varan inte finns med i filen, ska ett informativt och rättvisande meddelande skrivas ut. Om det lägsta priset för varan finns i flera olika butiker, räcker det att skriva ut en av butikerna.

Uppgift 13 (5 p)

Binärfilen **tal.bin** innehåller flera biljoner (10^{12}) flyttal (**float**). Samma tal råkar också finnas i textformat på filen **tal.txt**.

Skriv ett program som kollar att det verkligen är samma tal på båda filerna, i samma ordning! Skriv ut ett tydligt meddelande till användaren om hur det var.
