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

Tentamen i distanskursen

Programmering C

lördag 18 augusti 2018

Gäller som tentamen för:
DT104G Programmering C, provkod 0100

Hjälpmedel:	Ordbok för översättning.
Poängkrav:	Maximal poäng är 40. För godkänt betyg (3 respektive G) krävs 20 poäng.
Resultat och lösningar:	Meddelas via e-post senast lördag 8 september 2018.
Återlämning av tentor:	Efter att resultatet meddelats kan tentorna hämtas elektroniskt via Studentforum.
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. Gjorda antaganden får inte förändra den givna uppgiften.
- 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 *str
size_t fwrite(const void *ptr, size_t size, size_t nmemb, FI
```

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);
```

math.h

```
double sqrt(double x);
double pow(double x, double y);
```

För uppgifterna på tentan gäller: Om du ska använda något från en tidigare uppgift eller deluppgift, till exempel anropa en funktion från den tidigare uppgiften, behöver du inte skriva koden på nytt. Du får också göra uppgiften även om du inte gjort den tidigare uppgiften som den bygger på.

För uppgifterna 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 1 (5 p)

Vi ska skriva ett program som frågar efter två tal, och sen skriver ut resultatet av om man tar det ena talet upphöjt till det andra. Körexempel, med användarens inmatning understruken:

```
Ange basen: 2
Ange exponenten: 6
Resultat: 64
```

Så här blev programmet:

```
/* Ett program som räknar ut ett tal upphöjt till ett annat */

#include <stdio.h>;

int power(int base, int exponent) {
    int result = 0;
    for (i = 0; i < base; i--)
        result = result * base;
}

int main(void) {
    int basen;
    printf("Ange basen: ");
    scanf(&basen);
    printf("Ange exponenten: ");
    scanf("%f", &exponenten);
    int exponenten;
    power(basen, exponenten);
    int resultatet;
    resultatet = power;
    printf("Resultat: ", resultatet);
    return 0;
}
```

Tyvärr har det blivit ganska många fel i programmet. Tala om vilka fel det är, och visa vad det borde stå i stället!

Programmet finns även på sista sidan i tentan. Du kan göra rättelserna på den sidan, och sedan riva loss den och lämna in den tillsammans med dina övriga lösningar, så slipper du skriva av programkoden.

Uppgift 2 (1 p)

Vilka värden har följande C-uttryck?

a) `10 / 3 + 2`

b) `10 + 2 / 3`

c) `1 && 0`

d) `1 < 2 || 1 < 3`

Uppgift 3 (2 p)

a, **b** och **c** är heltalsvariabler. Ange värdet på a, b och c då följande kod har körts.

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

Uppgift 4 (2 p)

a och **b** är strängvariabler, definierade så här:

```
char a[17];
char b[17];
```

Skriv de kodrader som behövs för att byta plats på innehållet i dessa två variabler.

Uppgift 5 (4 p)

a) (2p) Skriv en funktion, **f**, som beräknar följande uttryck.

$$\frac{x^{y+1}}{1-x\sqrt{x^4-y^5}}$$

Funktionen ska ta de två flyttalen **x** och **y** som argument, och returnera uttryckets värde.

b) (2p) Skriv en **main**-funktion som läser in värden på **x** och **z**, beräknar uttryckets värde genom att anropa funktionen ovan, och sen skriver ut detta värde.

Uppgift 6 (5 p)

a) (3p)

Skriv en funktion **jamfor** som jämför två tal. Funktionen ska ta argument: dels de två talen, som är flyttal, och dels en textsträng som ska vara antingen "mindre", "större" eller "lika". Om textsträngen är "mindre", ska funktionen returnera sant om det första talet är mindre än det andra, annars falskt. Om textsträngen är "större", ska funktionen returnera sant om det första talet är större än det andra, annars falskt. Om textsträngen är "lika", ska funktionen returnera sant om det första talet är lika med det andra, annars falskt. Exempelvis ska anropet **jamfor(7, 8, "större")** ge ett falskt värde.

Om textsträngen är något annat än "mindre", "större" och "lika", ska programmet avslutas med ett felmeddelande.

b) (2p)

Skriv en **main**-funktion som låter användaren skriva in två flyttal, och sen anropar funktionen **jamfor** från uppgiften ovan för att ta reda på om de är lika. Om de är lika, ska **Lika** skrivas ut, annars **Inte lika**.

Scenario till de följande uppgifterna

Här är en grantopp med några kottar:

Vi ska lagra data om träd och kottar. En kotte har en längd och en vikt. Ett träd har en höjd, och högst tusen kottar.

Uppgift 7 (14 p)

a) (3p)

Skapa posttyperna **struct Kotte**, som representerar en kotte med längd och vikt, och **struct Trad**, som representerar ett träd och dess högst tusen kottar.

Välj själv om trädets kottar ska lagras i en array i trädposten, eller som en länkad lista.

b) (3p)

Skriv en funktion som heter **visa_trad**, och som visar data om ett träd, inklusive alla kottarnas data, enligt scenariot ovan.

Välj själv hur funktionshuvudet ska se ut, men trädet ska på något sätt överföras till **las_trad** från funktionen som anropade den.

c) (4p)

Skriv en funktion som heter **las_trad**, och som läser in data om ett träd, inklusive alla kottarna, enligt scenariot ovan. Funktionen ska skriva ut

lämpliga ledtexter, på standardutmatningen, och läsa in data från standardinmatningen (som normalt är kopplad till tangentbordet).

Välj själv hur funktionshuvudet ska se ut, men det inmatade trädet ska på något sätt överföras tillbaka till funktionen som anropade **las_trad**.

d) (2p)

Skriv en funktion som heter **kottvikt**, som returnerar den sammanlagda vikten av alla kottarna i ett träd.

Välj själv hur funktionshuvudet ska se ut, men trädet ska på något sätt överföras till **kottvikt** från funktionen som anropade den.

e) (2p)

Skriv en **main**-funktion som innehåller tre lokala variabler av typen **struct Trad**, som läser in data till dem med funktionen **las_trad**, och som till slut skriver ut de tre trädens sammanlagda kottvikt. Använd funktionen **kottvikt** för att beräkna varje träds kottvikt.

Uppgift 8 (7 p)

a) Skriv funktionen **spara_trad**, som sparar data om ett träd (enligt scenariot ovan) på en fil. Funktionen ska ta filnamnet som argument, plus att trädet på något sätt ska överföras till funktionen.

b) Skriv funktionen **hemta_trad**, som läser in data om ett träd (enligt scenariot ovan) från en fil. Funktionen ska ta filnamnet som argument, plus att trädet på något sätt ska returneras från funktionen.

c) Skriv en **main**-funktion som innehåller en lokal variabel av typen **struct Trad**, som hämtar in data till den med funktionen **hemta_trad**, och som därefter sparar trädet på en annan fil med funktionen **spara_trad**.

Lösningsblankett till uppgift 1

Tentamenskod:

```
/* Ett program som räknar ut ett tal upphöjt till ett annat */

#include <stdio.h>;

int power(int base, int exponent) {
    int result = 0;
    for (i = 0; i < base; i--)
        result = result * base;
}

int main(void) {
    int basen;
    printf("Ange basen: ");
    scanf(&basen);
    printf("Ange exponenten: ");
    scanf("%f", &exponenten);
    int exponenten;
    power(basen, exponenten);
    int resultatet;
    resultatet = power;
    printf("Resultat: ", resultatet);
    return 0;
}
```