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 2 mars 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 38. För godkänt betyg (3 respektive G) krävs 19 poäng.
Resultat och lösningar:	Meddelas via e-post eller på kursens hemsida, http://basen.oru.se/kurser/c/2012-2013-p2/, senast lördag 23 mars 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 (3 p)

Vad skrivs ut när följande C-program körs?

```
#include <stdio.h>

int plusett(int x) {
    return x + 1;
}

void loop(int varv) {
    int a;
    for (a = 0; a < varv; ++a)
        printf("*");
}

int main(void) {
    int a = 1 * 2 + 3 * 4;
    int b = 1 / 2;
    float c = 1 / 2;
    printf("a = %d\n", a);
    loop(1);
    printf("b = %d\n", b);
    loop(plusett(1) + plusett(2));
    printf("c = %f\n", c);
    return 0;
}
```

Uppgift 2 (5 p)

Vi ska ut och resa, och ska packa våra resväskor. Flygbolaget har en viss maxvikt för varje väska, och nu behöver vi ett program som hjälper oss att räkna ut vad väskan väger.

Skriv ett komplett C-program (med **#include** och allt) som frågar efter maxvikten och vad själva väskan väger, och sen frågar efter enskilda sakers vikt ända tills man avslutar genom att mata in vikten noll. Saker som inte ryms inom den tillåtna vikten läggs inte i väskan. Avslutningsvis ska totalvikten skrivas ut. Alla vikter är flyttal.

Ett körexempel, med användarens inmatning understruken:

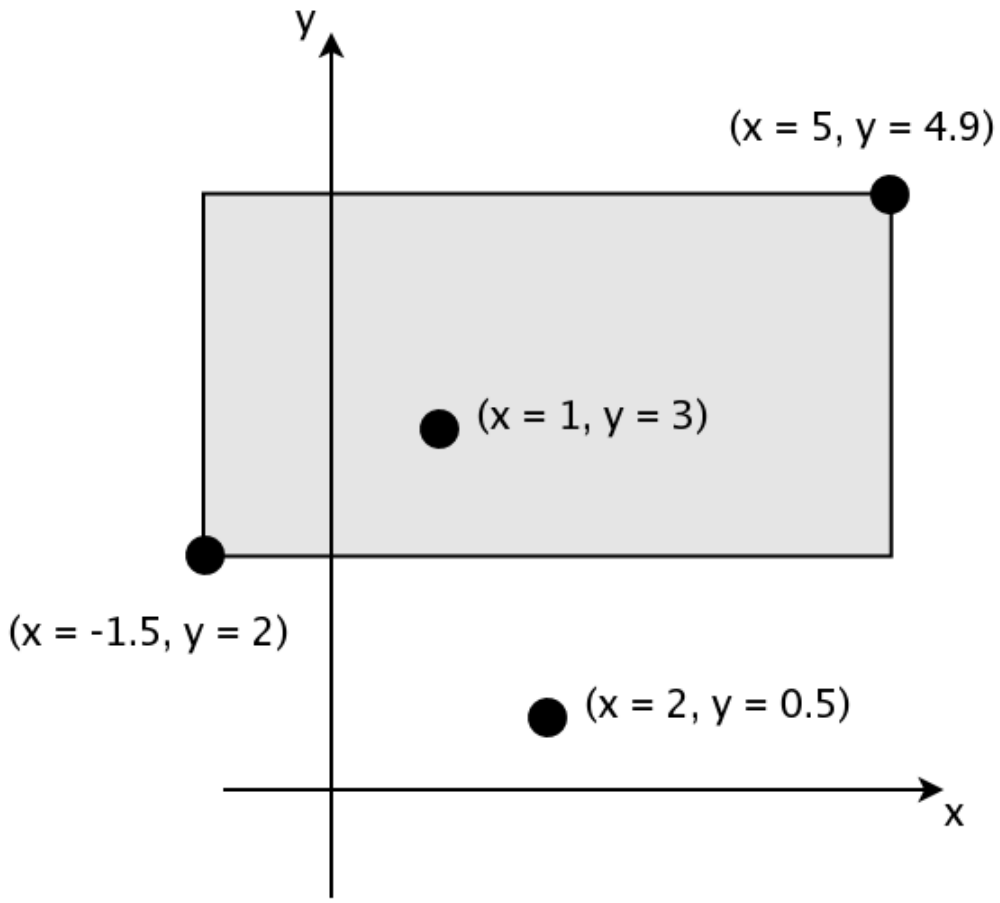
```
Ange högsta tillåtna totalvikt: 20
Ange själva väskans vikt: 4
Ange sakernas vikter. Avsluta med 0.
Vad väger nästa sak? 2
Ok, lägg i den.
Vad väger nästa sak? 3.5
Ok, lägg i den.
Vad väger nästa sak? 16
Nej, det blir för tungt!
Vad väger nästa sak? 6
Ok, lägg i den.
Vad väger nästa sak? 0.2
Ok, lägg i den.
Vad väger nästa sak? 0
Totalvikt: 15.70
```

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.

I den här och alla andra uppgifter på tentan gäller:
Man kan strunta i detaljer som bara behövs just när man utvecklar konsolprogram i Visual Studio, som konstiga teckenkoder för ÅÄÖ, och att fönstret med programkörningen försvinner när programmet avslutas.

Uppgift 3 (1 p)

Här är ett koordinatsystem med en rektangel och några punkter, som ni kanske känner igen från en av inlämningsuppgifterna.



Nu ska vi arbeta med rektanglar. Skapa därför datatypen **struct Rektangel**, som beskriver en rektangel. Välj själv hur posttypen ska se ut, men ett sätt är att lagra x- och y-koordinaterna för två hörn, som på bilden.

Uppgift 4 (1 p)

Definiera en variabel av typen **struct Rektangel** och initiera den med data om rektangeln i figuren i uppgiften ovan.

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 5 (2 p)

Skriv funktionen **area**. Funktionen ska ta en rektangel som argument och returnera dess area. (Arean fås genom att multiplicera höjden med bredden.)

Uppgift 6 (2 p)

Skriv funktionen **mindre**, som jämför två rektanglar. Funktionen ska ta två rektanglar som argument, och om den första har mindre area än den andra ska funktionen returnera ett sant värde. Annars ska den returnera ett falskt värde. Funktionen **mindre** ska använda sig av funktionen **area**.

Uppgift 7 (3 p)

Skriv en funktion som heter **total_area**, som tar en array med rektangel-poster som argument, tillsammans med ett heltalsargument som anger antalet rektanglar i arrayen, och returnerar summan av rektanglarnas area. (Vi behöver inte göra något särskilt för att hantera överlappande rektanglar.)

Uppgift 8 (5 p)

Vi vill provköra funktionerna **area**, **mindre** och **total_area** från uppgifterna ovan. Skriv därför en **main**-funktion, som anropar dem med lämpliga argument, och skriver ut ett tydligt felmeddelande om någon av dem inte ger det förväntade resultatet.

Vi ska bara anropa varje funktion en gång, så det finns alltså bara ett enda testfall per funktion. Försök göra dessa testfall så bra som möjligt!

Uppgift 9 (4 p)

Skriv ett program som läser in tio reella tal till en array, och därefter skriver ut det största talet, det minsta talet, och talens summa.

Uppgift 10 (4 p)

Skriv ett program som från tangentbordet läser in ett antal reella tal, ända tills man avslutar genom att mata in talet noll, och skriver dessa tal på en fil. Välj själv om det ska vara en text- eller binärfil. Om filen inte går att öppna, ska ett felmeddelande skrivas ut, och programmet ska avslutas.

Uppgift 11 (4 p)

Skriv ett program som läser filen från uppgiften ovan, och skriver ut hur många tal det var.

Uppgift 12 (4 p)

Skriv ett program som slumpar fram en fil som kan läsas av programmet i uppgiften ovan. Programmet ska slumpa fram både antalet tal och de enskilda tal som skrivs på filen.

Ledtråd: Funktionen **rand** ger ett slumpmässigt heltal, och funktionen **srand** initierar slumpsekvensen.