Erfarenheter från provbedömning med skriftlig feedback istället för poäng

Filip Sebek, IDT, Mälardalens Högskola i Västerås

2009-08-06

1 Introduktion

I prov eller bedömningssituationer där bedömningen sker på en uppsats eller essä som skrivits är det ganska vanligt med skriftlig kommentar som motivering till ett betyg men även som en återkoppling till studenten för att denna ska kunna förbättra sin kunskap eller kompetens inför framtida omtentor, studier eller arbetsliv. Inom andra typer av prov ges däremot poäng som oftast ska spegla hur stor del av uppgiften som är löst korrekt och som ibland kan kompletteras med en kommentar på vad som är fel (eller rätt). Med gränser för vad den sammanlagda summan av poäng som erhållits från provet kan sedan ett betyg tas fram. Frågeställningen vid en traditionell poängsättning av ett antal uppgifter på ett prov är flera;

- Hur viktas poängen i uppgifterna mot olika delar och innehåll av kursen?
- Hur viktas poäng mot uppgifternas svårighet?
- Hur väl speglar en viss poäng ett betyg som i sin tur ska spegla en viss kompetens?
- Hur tolkar studenten poängsättningen och det totala resultatet?
- Hur tolkar examinatorn och andra lärare poängsättningen och det totala resultatet?
- Vilken feedback ger poängsatta prov till studenten?

Ett sätt att undvika några av ovanstående frågeställningar och problem är att inte poängsätta de examinerande uppgifterna utan istället ge kommentarer likt uppsats-bedömningen på vad studenten har gjort rätt och fel (Sebek 2005). Med Bolognaprocessens införande av mål i bland annat kursplaner blir det naturligt att examinationen sker mot dem istället för mot kursens syfte eller innehåll.

2 Genomförandet

Kursen CDT106 ”Programmeringsteknik med C” är en grundläggande kurs i programmering som alla datastudenter läser i början av sin utbildning. Examinerande moment är laborationsuppgifter och en ordinär skriftlig tentamen med papper och penna. Bästa sättet att lära sig och att visa att man kan programmera är otvivelaktigt att lösa de programmeringsuppgifter i form av problem som finns i kursens laborativa del. Erfarenhetsmässigt har det dock visat sig att en mindre andel studenter inte löser uppgifterna själva eller förstår hur man löser ett problem, och även om de tangerar på rätt sida om gränsen för akademiskt fusk så är det enklast för både lärare och student att visa vad man kan under de mera kontrollerade former som en enskild tentamen erbjuder. En skriftlig tentamen i programmering brukar vanligtvis innehålla ett antal problem som ska lösas med att skriva hela eller delar av ett datorprogram av varierad svårighetsgrad och ibland med instruktioner på vilka komponenter som ska ingå för att visa vilka delar inom området som behärskas. Studentens lösningsförslags bedöms och poängsätts ofta efter en i förväg konstruerad rättningsmall för att garantera en likvärdig bedömning. Förutom de redan nämnda problemen i det inledande stycket så bör beaktande även ske på att poängsättningen kan innebära att studenten blir godkänd trots att kursens lärandemål inte är uppfyllda. Studenten kan erhålla poäng för perifera delar av kursinnehållet trots att lärandemålet inte täcks på ett tillfredställande sätt.

Genom att konstruera tentamen med ett antal olika uppgifter som täcker lärandemålen i första hand så kan man försäkra sig om att provet testar rätt saker – har en hög innehållsvaliditet (Wolming 1998). I samband med konstruktionen så är det viktigt att även notera i rättningsmallen vad det är som ska bedömas. På detta sätt kan man kontrollera tillbaka att uppgiftens formulering ligger i linje med det som avses testas. Lärandemålen i kursplanen ska vara minimikrav på att få bli godkänd varför någon viktning eller poängsättning inte är nödvändig. Istället kan man beskriva i klartext på studentens lösningsförslag vad det är som är bra och dåligt i lösningsförslaget.

I den kurs som genomfördes under hösten 2007 fanns flera godkända betygssteg (U,3,4,5) varför ytterligare kriterier var nödvändiga att formuleras för att i den samlade bedömningen kunna ge rätt betyg. I kursPM[1] (se bilaga A) för kursen beskrivs i allmänna ordalag vad som krävs för de olika betygen 3,4 och 5 och därefter rent konkret *vilka* och *hur många* av momenten som måste vara godkända för at få ett visst betyg. Det finns alltså en kvantitativ del som bestämmer betyget men de fundamentala delarna som beskrivs i lärandemålen måste vara godkända innan ett högre betyg än 3 kan övervägas.

3 Erfarenheter

Det kan vid första anblicken te sig vara ett enormt merarbete att vid rättning och bedömning behöva skriva en massa text jämfört med att sätta en siffra, men faktum är att görs det vid en dator så kan mycket ”klipp-och-klistra” användas – till skillnad från en röd kulspetspenna. Det riktigt svåra är att skriva bra uppgifter som verkligen visar att studenten kan visa att denne behärskar en viss teknik. I detta pilotförsök med att examinera mot lärandemål så examinerades alla lärandemål vid tentamenstillfället vilket så här i efterhand inte är nödvändigt. Man skulle mycket väl kunna flytta de delar som examinerar mot de högre betygen till t.ex. en projektuppgift. Man skulle också kunna behålla poängsystemet och lägga in kriteriet att man måste ha minst hälften av den maximala poängen på vart och en av alla ”grunduppgifter” – konstruktionerna och modellerna är många.
Som lärare kände jag mycket bekväm med att rätta på detta sätt och jag kände större säkerhet i att alla hade fått en bra bedömning än vid tidigare tentamenstillfällen då jag mycket konkret kunde motivera varför jag godkände eller underkände en student och kunde hänvisa till kursplanens mål.

3.1 Studenternas åsikter

Enligt den anonyma kursutvärdering som utfördes efter tentamen så tyckte samtliga att bedömningsformen "skriftligt omdöme utan poängsättning" så som den var utformad var bättre eller mycket bättre än traditionell poängsättning och att de hade fått rätt slutbetyg. Samtliga hade också upplevt det bedömningsprotokoll de fått hade gett dem god eller mycket god förståelse för varför de fått det betyg de fått. Ingen student ansåg att metoden var mer subjektiv än traditionell poängsättning även om flera ansåg sig inte kunna göra den bedömningen. Enkelt sammanfattat så var studenterna nöjda med det nya bedömningsförfarandet.

[1] Av vissa även kallad studiehandledning eller kursinformation.

4 Teknisk beskrivning på genomförandet

I kursen hade WebCT/Blackboard använts för att dokumentera avklarade moment såsom laborationer, duggor och projekt. Genom att lista hela klassen som en tabell så kunde denna markeras med musen, kopieras och klistras in i Excel för enklare redigering. Förutom de olika resultaten så hamnar namn och personnummer i egna kolumner. Arket kan sedan sparas som en fil. Därefter importeras excelfilen in i databasprogrammet MS Access som steg för steg frågar hur den nya databastabellen ska lagras. En god idé är att ha personnumret som primärnyckel eftersom den är unik. När tabellen väl är skapad så kan nya fält läggas till. I det här fallet valdes ett fält för varje uppgift på tentamen samt några ytterligare fält för slutbetyg och kommentarer. Allt fylldes i via formulär i MS Access och varje student fick sedan sitt personliga bedömningsprotokoll med kommentarer bifogat tentamen. Även om vissa kommentarer blev identiska hos olika studenter, då de löst uppgiften på liknande sätt) så blev kombinationen av alla kommentarer helt unik för varje student. Några noteringar i studentens egna lösningsförslag gjordes aldrig utan allt skedde i databasen. Examinatorns noteringar blev därför mycket läsliga och lösningsförslag med många fel behövde inte rödmålas vilket kan upplevas som jobbigt att hantera för en student som precis insett att denne blivit underkänd på sin första universitetstentamen. Några exempel på bedömningsprotokoll finns i Bilaga C

5 Slutsatser

För den som vill och orkar göra sina tentor på ett annat sätt så är den här modellen bra potential att bli ännu bättre när det finns mera erfarenheter kring dem. Filer, databaser, mallar och utvärderingar finns i sin helhet i det elektroniska kurspärmsarkivet (f.d. coursedoc).

6 Referenser

Sebek 2005 *“Written critique instead of a score”*, ASEE/IEEE 2nd CeTUSS Workshop, Uppsala, Sverige, april 2005

Wolming 1998 “Validitet – Ett traditionellt begrepp i modern tillämpning”, Pedagogisk forskning I Sverige 1998, Årg 3, nr 2, s81-103, ISSN 1401-6788

Kursinformation, CDT106 Programmeringsteknik med C

CDT106, Ingenjörsstudenter i datateknik och robotik
dagtid, 25%fart, Läsperiod I & II hösten 2007
Institutionen för Datavetenskap och Elektronik (IDE), Mälardalens Högskola (MDH)

Välkommen!

Du kommer under kursens gång att lära dig grunderna i programmeringsteknik och att lösa mindre uppgifter genom att skriva program i programmeringsspråket C. Att programmera handlar om en färdighet som du bara kan lära dig genom att öva och skriva egna program. Att läsa om och studera hur andra skriver program kan ge en viss inblick i processen, men det enda sättet att lära sig programmera är att själv skriva egna program.
Laborationsuppgifterna som du ska lösa och redovisa är därför centrala i kursen, men du behöver också öva utöver dessa uppgifter. Undervisningen i form av föreläsningar och övningar i lektionssal, liksom handledning av uppgifterna är ett stöd för dig i ditt lärande. Att kunna programmera handlar till viss del att förstå hur program kan skrivas, men som i alla konstruktiva processer så handlar det främst om att komma på. Om du inte programmerat tidigare så kommer du att upptäcka att programmering är till stor del en skapande konst som är väldigt rolig och spännande bara man har tagit sig över den första tunga uppförsbacken.

Lärare

Filip Sebek kan hittas i rum U3-113 eller på telefon 021-103113, E-post skickas till `filip.sebek@mdh.se`
Marcus Blom i rum U3-014 är laborationsassistent och har e-post `marcus.blom@mdh.se`

Mål

Enligt kursplanen har kursen följande lärandemål som dessutom ska ses som minimikrav för att uppfylla förkunskapskraven inför kommande kurser: Studenten ska efter genomgången kurs

- kunna skriva ett strukturerat C-program som löser ett enkelt problem.
- ha förmåga att kunna avgöra vilka grundläggande språkelement som bör användas för att lösa ett visst givet problem.
- kunna läsa programkod, skriven i C, och med egna ord beskriva vad programmet utför.

Kursen är grundläggande för dig som läser på ett ingenjörsprogram på IDE och är i många fall ett absolut förkunskapskrav inför andra kurser som finns på programmet.

Undervisning

Kursen består av tio **föreläsningar**, sex handledda **övningsstillfällen**, **laborationer** och en **projektuppgift**.
Någon ”obligatorisk närvaro” finns inte, men man gör klokt i att ta reda på vad som tas upp på föreläsningar och övningar eftersom litteraturen inte fullständigt speglar betoningen och vikten på de olika delarna i kursen.
Föreläsningarna är vanliga lektioner med genomgång av kursinnehållet där man förväntas ha studerat igenom materialet översiktligt innan så att man kan ställa relevanta frågor. Den schemalagda undervisningen räcker absolut inte till att vara heltäckande – det blir ett urval av sådant som är viktigt och centralt i kursen samt sådant som läraren erfarenhetsmässigt vet är svårt. Lätt eller mer perifert stoff måste man läsa in själv.
Övningar är schemalagda tillfällen där man som student mera aktivt ägnar sig åt problemlösning och övning med handledning av lärare. Uppgifter brukar lämnas ut i förväg där man under själva lektionen ägnar tid att lösa och jämföra olika lösningsförslag.
Laborationer är uppgifter som löses vid datorer och redovisas och godkännes av labass vid respektive labtillfälle. Labbar i denna kurs är till skillnad från föreläsningar och övningar ett examinerande moment. Vid uppropet väljes labgrupp och man förväntas följa det schemat. Om man skulle vara förhindrad vid något tillfälle kan man kolla om det finns ledig labdator vid annat tillfälle. Viktigt: Det är tillåtet att göra labbarna på annan plats och redovisa efter hand men *att göra många eller alla labbar i ett svep och därefter vilja redovisa allt vid ett tillfälle* ***accepteras inte*** *alls. Redovisningen* ***skall*** *ske kontinuerligt.* **OBS**ervera att man behöver i allmänhet mer tid än de schemalagda 4 timmarna till varje laboration för att hinna lösa och redovisa uppgifterna. Schemalagda timmar är *handledningstid*. Räkna med att du behöver förbereda och läsa in dig.

Deadline

Labbar och projektuppgiften ska normalt sett vara klara vid det sista redovisningtillfället vecka 51. Därefter kommer man kunna boka tid för rester (endast) vecka 3, 13, och 24 och i omtentaveckorna i augusti 2008. Om du inte blir klar innan augusti 2008 så kan du behöva göra om *hela* laborationsserien och/eller en annan projektuppgift. Observera att många kurser kräver att du är helt klar med denna grundläggande kurs för att få bli registrerad på dem, så beroende på vilken utbildning man vill ha bör man noga prioritera sina resurser om de inte räcker till allt.

Kurslitteratur

Det finns en uppsjö av böcker i C, och många av dem duger till denna kurs. Alla referenser kommer dock att utgå från **"Problem solving with C"** av **Jacqueline A. Jones & Keith Harrow**. Denna bok finns i princip endast tillgänglig på Akademibokhandeln på högskolan i Västerås eftersom den är speciellt upptryckt för denna kurs. Bokens största förtjänst är att den vänder sig till totala nybörjare i programmering och beskriver hur själva problemlösnings*processen* går till. De flesta andra böcker brukar bara slänga upp svaret på lösningen och sedan argumentera och förklara sin lösning. För vissa kan detta dock vara tillräckligt om man har skrivit egna program tidigare. En alternativ bok på svenska är den nya boken "C genom ett nyckelhål" av Håkan Strömberg som är den enda boken på svenska som vi kan rekommendera. Alla andra svenska böcker vill vi avråda från. Valet av bok står alltså mellan god pedagogik eller ett lite mera lättillgängligt språk. Inget hindrar från att man skaffar båda, köper dem tillsammans med en kompis eller köper begagnat.

Kurshemsida

Som informationskanal används kursadministrationsverktyget WebCT (Web Course Tool) som ligger på `http://webct.mdh.se` (obs inget "www"). Här loggar du in och kan ta del av nyheter, allmän information, boka redovisningstider, se vilka moment du är godkänd på m.m. Du loggar in med samma användarID och lösenord som du använder till studentportalen eller din e-post på MDH.
Det är mycket viktigt att du minst en gång mellan varje schemalagt tillfälle (helst oftare) loggar in på WebCT. Förändringar och akut information distribueras oftast endast via denna kanal.

Om fusk

Det är fel att fuska. Det ligger i allas intresse att fusk förhindras såväl vid tentamen som vid laborationer. Med fusk i utbildningssammanhang menas att man vid examinerande moment försöker utge sig för att ha gjort momentet när man inte har det. Några vanliga exempel på fusk är plagiering; t.ex. kopiering, parafrasering och översättning, otillåtet samarbete, att åka snålskjuts på andras arbete, fabricering, samt otillåtna hjälpmedel.

För att undvika att oavsiktligt hamna i fusksammanhang följ dessa enkla riktlinjer;

- Läs instruktionen noga till examinerade uppgifter och kursanvisningar – inte bara uppgiften.
- Om uppgiften ska lösas individuellt – lös den på samma sätt som om du skrev en salstenta. Om uppgiften ska lösas i grupper om två, så får inte två grupper (=4 personer) samarbeta.
- Beskriv noga hur du har löst uppgiften; verktyg, metod, hjälp, källor – allt ska nämnas i förhand. Om det visar sig att du tagit till otillåtna hjälpmedel blir du i värsta fall bara underkänd vilket är en oerhörd skillnad jämfört med en fuskanklagelse.
- Namnen på omslaget till en inlämnad uppgift är de, och endast de, som har varit med om att lösa uppgiften.

Du får gärna diskutera kurslitteraturen, material från föreläsningar och generella principer med andra, men det innebär inte att du kan ta otillåten hjälp för att lösa själva uppgiften. Att hjälpa någon att fuska ses oftast som lika allvarligt som själva fuskandet och kan också leda till disciplinära åtgärder.

I händelse att fusk misstänks kommer detta att anmälas till rektor för handläggande av disciplinnämnden. Om fusk kan styrkas kan man bli varnad eller i allvarliga fall bli avstängd från undervisningen under en tid.

Du är skyldig att känna till och följa de regler som finns kring fusk. Okunskap är ingen ursäkt eller förmildrande omständighet! Om du inte förstår eller är osäker kring något i ovanstående beskrivningar, läs mer på http://www.mdh.se/ide/utbildning/cheating/ eller prata med din lärare.

Examination, Bonus & Betygsystem

För godkänt på kursen krävs det godkänd tentamen, vilket ger 3 högskolepoäng (hp), godkända laborationer, 3 hp samt godkänd projektuppgift, 1½ hp.
Första ordinarie tentamenstillfälle är 080107 och för att få skriva måste man registrera sig vilket företrädesvis ska göras via MDH:s studentportal ca 3-4 veckor innan tentan ges. Registreringen måste göras minst 6 arbetsdagar innan tentamensdagen. Tänk på att det är många arbetsdagar som försvinner i samband med julhelgerna. Är man inte registrerad så får man inte skriva – *inga* undantag ges för sena anmälningar! För att få tentera skall giltig legitimation och betalt kåravgiftskvitto uppvisas vid tentamenstillfället. På tentamen i denna kurs får man ha kursboken med sig som hjälpmedel.

För att få godkänt på kursen så måste minimikraven som står beskrivna i målet (se ovan) med kursen vara uppfyllda.

Betygen är målrelaterade

- Betyget 3 (minimikrav)
    - kunna skriva ett strukturerat C-program som löser ett enkelt problem.
    - ha förmåga att kunna avgöra vilka grundläggande språkelement som bör användas för att lösa ett visst givet problem.
    - kunna läsa programkod, skriven i C, och med egna ord beskriva vad programmet utför.

- Betyget 4 (tillägg till ovanstående)
    - kunna analysera ett program och ge förslag på alternativa lösningar på hur problemet kan lösas
    - kunna tillämpa externa funktionsbibliotek i egna program genom att läsa dess dokumentation

- Betyget 5 (tillägg till ovanstående)
    - kunna analysera ett program med avseende på felsökning, återanvändbarhet, underhåll m.m.

Vid tentamensrättningen upprättas ett protokoll med checklista för att redovisa uppnådd betygsnivå.

Ungefär halvvägs in i kursen ges en ”dugga” som är ett icke-obligatoriskt prov som ger tillgodoräknanden till ordinarie tentamen. Är man klar med laborationerna eller projektuppgiften så ger även det tillgodoräknanden till tentamen. Bonussystemet är till för att uppmuntra kursdeltagarna att bli klara med alla moment innan slutprovet – något som brukar ha givit tillräckliga kunskaper för att klara tentan.

Preliminära bedömningsaspekter

Alla uppgifter på tentamen innehåller några olika element av kursinnehållet. Den samlade bedömningen av detta element i relevanta uppgifter ligger till grund för bedömningen av kunskaper och insikter. För högre betyg viktas även kommentarer med avseende på målkriterierna (se föregående avsnitt) för de olika betygen in.
Systemet är nytt i och med införandet av mål i kursplanen fr.o.m. höstterminen 2007 varför gränsvärdena i nedanstående tabell är preliminära då ingen praxis har hunnit tillämpas

Betyg	Grundläggande moment	Kriterier på mer avancerade moment
3	Alla.	2 G
4	Alla	4 G
5	Alla	8 G

Gränsvärdena kommer vid en eventuell justering givetvis vara lika för alla bedömda kursdeltagare.

Grundläggande moment (alla ska vara godkända)

#	Kunskap &	insikt	Teori & praktik (omdöme + kommentar)
1	Variabler	Deklaration & tilldelning	
2		Typomvandling	
3	Selektion		
4	Iteration	Grundläggande	
5		Nästlad sats iteration	
6		Nästlad sats selektion	
7	Arrayer	Deklaration & tilldelning	
8	Funktioner	Modularisering	
9		Anrop Call-by-value	
10		Returvärde	

Mer avancerade moment

#	Kunskap &	insikt	Teori & praktik (omdöme + kommentar)
11	Pekare	Grundläggande	
12	Strängar	Manipulering	
13	Funktioner	Call-by-reference	
14		Använda ”externa”	
15	Filhantering	Grundläggande	
16		Mer avancerad	
17	Strukturer	Deklaration	
18		Användning av	
19	Dynamisk minneshantering	Grundläggande	
20		Länkade strukturer	
21	Klara labbar		
22	Klart projekt		
23	Dugga		
24	Pluslaborationr		

Tentamen i kursen

Programmeringsteknik med C

CDT106 & CD5020

Tid: Måndagen 7/1-2008 08:30-13:30

Ansvarig lärare: Filip Sebek, tfn:021-103113
Besöker skrivsalen ca 10:00 och 12:00

Hjälpmedel *En* bok – vilken som helst.
Noteringar i boken är tillåtna men inga lösa blad.

Om godkänt och högre betyg

- För godkänt krävs godkända G-uppgifter samt två A-moment (varje A-uppgift på denna tentamen innehåller minst två A-moment). Godkända laborationer, godkänd projektuppgift eller plus-uppgift motsvarar ett A-moment om de är godkänt utförda innan första tentamenstillfället.
- En uppgift måste inte vara helt rätt för att bedömas som godkänd. Mindre fel accepteras!
- En del uppgifter är till för att avgöra för högre betyg än betyget 3. Om du nöjer dig med betyget 3 så behöver du inte lösa dessa uppgifter.
- För att erhålla ett högre betyg så måste majoriteten av dina lösningar i denna tentamen vara så generellt skrivna att de på ett enkelt sätt kan förändras eller ev. kunna överföras till andra programmeringsprojekt. Kravet är alltså inte att bara lösa uppgiften som sådan utan det ska göras ”med finess” vilket innebär funktions-nedbrytning, val av bra variabel- och funktionsnamn, användandet av definierade konstanter samt att konstruera för enkel felsökning, läsbarhet och testbarhet. Kravet på kommenterad kod är dessutom hög – men kommentera inte självklarheter! ”Finess” betyder alltså inte brutalt komplexa lösningar sammanfattade i en enda okommenterad sats.
- Alla A-uppgifter räknas som A-uppgifter, dvs du kan välja att lösa en sådan uppgift för att förvärva A-moment för att nå Godkänt-nivån.. Däremot om man vill ha högre betyg så måste man även klara av högre-betygsuppgifterna. Med andra ord så är höja-betygsuppgifterna lite mera värda.

Viktiga instruktioner

- Max en uppgift per inlämnat blad.
- Alla inlämnade blad skall förses med namn och personnummer.
- Oläsbara lösningsförslag eller tvetydiga lösningar ger inte godkänt.
- Globala variabler och hoppsater (goto) ger stora negativa avdrag i bedömningen.
- Referera inte mellan olika uppgifter.
- Lösningsförslagen måste inte vara helt säkra i betydelsen att inmatade värden testas så de är korrekta eller rimliga, såvida detta inte står explicit i uppgiften

- Läs igenom uppgifterna noga – svara på frågan och inget annat.
- Fyll i försättsbladet ordentligt – Kolla särskilt ”Antal inlämnade blad”
- Kom ihåg
    - delvis lösta uppgifter bedöms också.
    - Kan du inte skriva i C – skriv i pseudokod. Du måste dock beskriva HUR du löser uppgiften – inte vad du vill lösa för det står ju redan i uppgiften.

Uppgift 1 – G-typ (motsvarar del 1 på duggan)

OBS! Om du har fått godkänt på del 1 på duggan som gavs under höstterminen 2007, så ska du inte lösa denna uppgift.

Ekvationen $x^2 + y^2 = r^2$ representerar en cirkel med linjen på {x,y}-koordinater, centrum i origo och radien *r*.
Skriv ett **C-program** som läser in *r* från tangentbordet. Därefter ska programmet läsa in koordinater från tangentbordet och skriva ut på skärmen vilka av koordinaterna som befinner sig i eller på cirkeln. Programmet ska läsa in nya koordinater ända tills användaren matar in koordinaterna x=0 och y=0 (origo ligger ju alltid i cirkeln, så informationen är överflödig men inte fel)

Exempel på körning:

```
Radie? 4
X? 3.0
Y? 1.02
Punkten ligger i eller på cirkeln
X? 2.05
Y? 10.75
Punkten ligger utanför cirkeln
X? 4.30
Y? 0.52
Punkten ligger utanför cirkeln
X? 1.75
Y? -2.0
Punkten ligger i eller på cirkeln
X? 0
Y? 0
Punkten ligger i eller på cirkeln
Programmet avslutas
```

Uppgift 2 – G-typ (motsvarar del 2 på duggan)

OBS! Om du har fått godkänt på del 2 i duggan som gavs under höstterminen 2007, så ska du inte lösa denna uppgift

Du ska göra ett **program** som frågar efter två tal som kan innehålla en eller flera decimaler. Därefter ska programmet anropa en **funktion** som du också ska göra. Denna funktion ska returnera det största av talen. Huvudprogrammet ska därefter skriva ut bägge talen och skriva ut vilket det största talet är.

Uppgift 3 – G-typ

Vad skriver följande kodsnutt - exakt - ut på skärmen

```
#include <stdio.h>
#define MAX 7
#define STAR '*'
#define BLANK ' '
int main(void)
{
  int i, j;
  for (i=0; i<MAX; i++)
  {
    for (j=0;j<i;j++)
      printf("%c", BLANK);
    for (j=i; j<MAX-i;j++)
      printf("%c", STAR);
    printf("\n");
  }
  return 0;
}
```

Du får gärna ta med text/skisser, som visar delresultat av ditt svar – en förutsättning för att du skall kunna bedömas om du gjort något mindre fel.

Uppgift 4 – G-typ & A-typ

Skriv ett **c-program** som ska läsa in två vektorer (samma sak som lista och array) av heltalstyp från användaren via tangentbordet. Längden på vektorerna definieras av LENGTH (som är definierad som prekompilator-direktivet #define) Programmet ska *därefter* avgöra vilken av vektorerna som innehåller det största elementet (skalär).

Exempel 1:
1: [1,5,3,3,2,9,1,1,**1234**,0]
2: [100,200,300,200,200,100,0,0,0,0]
→
"Vektor 1 innehåller det största värdet"

Exempel 2:
1: [23,23,**108**,0,0,0,0,0,0,0]
2: [29,26,0,**108**,0,0,0,0,0,6]
→
"Båda vektorerna har lika stort maxvärde!"

Om du vill kunna tillgodoräkna dig A-moment så måste programmet vara uppbyggt kring nedanstående funktionerna som du definierar, deklarerar och använder från main(). Tar du bara med en av funktionerna så ger det enstaka A-moment.

- **`readVector`** som läser in värden till en vektor via tangentbordet
- **`biggestInVector`** som returnera största värdet i en vektor som skickas in som inargument

Uppgift 5 - A-typ (för högre betyg)

Lös hela uppgift 4 igen fast på ett alternativt sätt. Du ska med andra ord visa att du behärskar flera olika tekniker på att kunna lösa samma sak.
Några exempel:

- Använd minst en annan konstruktioner i loopar än de du använde i uppgift 4.
- Om du använde dig av
    - indexering av vektorerna i uppgift 4 så ska du nu använda dig av pekarstegring
    - pekarstegring av vektorerna i uppgift 4 så ska du nu använda dig av indexering

Uppgift 6 – A-typ

Du ska göra ett **program** som kan hantera en tidpunkt. Tidpunkten ska lagras i programmet som en enda variabel formad som en struct, där variabeln har tre delar, ett värde för timmar (0-23), ett för minuter (0-59) och ett för sekunder (0-59).
Gör ett program uppbyggt kring en enkel meny i fyra delar där du kan välja mellan att mata in nytt värde på tidpunkten, lägga till ett antal sekunder till tidpunkten (mellan 0 och 200), skriva ut aktuellt värde på tidpunkten och avsluta programmet. Vid val lägga till sekunder skall en funktion anropas där den nya tidpunkten räknas ut. Du måste även redovisa en separat **algoritmbeskrivning** (t.ex. pseudokod) av funktionen som lägger till ett antal sekunder – kommentarer i koden räcker inte.
Ditt program får (men måste inte) kunna hantera större tidstillägg än ovan nämnda och behöver heller inte kunna hantera negativa tidstillägg.

Uppgift 7 – A-typ

Skriv en **funktion** - med följande prototyp: `void cpyline(char *filename);`
. . . som öppnar och kopierar första raden i textfilen filename och sedan ”klistrar” in denna rad sist i filen.

Exempel: Om innehållet i textfilen med filnamnet ”filename” är följande:

```
adam
bertil
caesar
```

. . . så skall följaktligen detta förändras till. . .

```
adam
bertil
caesar
adam
```

. . . när funktionen void cpyline(char *filename) har gjort sitt jobb.

OBS! Det är tillåtet att skapa en helt ny fil med samma innehåll som i den nedre rutan och då ska filnamnet vara detsamma fast med tillägget ”2”, Exempel: filename innehåller strängen ”nisseslista.txt” så blir det nya filnamnet på filen med det modifierade filinnehållet ”nisseslista.txt2”

Uppgift 8 – A-typ (för högre betyg)

Skriv en **funktion** som har som inargument en textsträng som innehåller ett namn. Om namnet är mer än 20 tecken så ska det till strängen läggas till texten ”är ett långt namn” och om det är kortare så ska det istället läggas till ”är ett kort namn”. Funktionen skulle exempelvis kunna användas på följande sätt

```
#include <stdio.h>
#include <string.h>

void addTail(char *str);  /* du ska implementera
                             definitionen till denna funktion */
int main(void)
{
  char namnet[100];
  printf("Mata in ett namn:");
  gets(namnet);
  while(namnet[0]!='q') {
    addTail(namnet);
    printf("%s\n",namnet);
    printf("Mata in ett namn:");
    gets(namnet);
  }
  return 0;
}
```

Och ger då följande körning (fetmarkerat är inmatningar av användaren)

```
Mata in ett namn:Filip Sjöbäck
Filip Sjöbäck är ett kort namn.
Mata in ett namn:Signe Inge Blockflöjtsson
Signe Inge Blockflöjtsson är ett långt namn.
Mata in ett namn:q
```

För att få godkänt på denna uppgift MÅSTE du använda dig av funktionen strcat() dokumenterad enligt nedan

NAME

strcat - concatenate two strings

SYNOPSIS

```
#include <string.h>
char *strcat(char *s1, const char *s2);
```

DESCRIPTION

The *strcat()* function appends a copy of the string pointed to by *s2* (including the terminating null byte) to the end of the string pointed to by *s1*. The initial byte of *s2* overwrites the null byte at the end of *s1*.

RETURN VALUE

The *strcat()* function returns *s1*; no return value is reserved to indicate an error

Bedömningsunderlag för Tentamen CDT106

Uppg 1 +	G - Dugga
Uppg 1 -	
Uppg 2 +	G - Dugga
Uppg 2 -	
Uppg 3 +	Korrekt utskrift vilket indikerar förståelse för programflöde, komplexa nästlade loopar och förmåga att sätta sig in i kod skriven av andra.
Uppg 3 -	
Uppg 4 +	Programmet är välskrivet, meningsfullt kommenterat och anropar funktionerna enligt specifikationen i uppgiften. Hanterar arryer med indexering på ett korrekt sätt både vid tilldelning och I/O
Uppg 4 -	
Uppg 5 +	Har gjort i princip alla variationer som går att göra - vissa blir då väldigt omständiga; som att t.ex. returnera en pekare men det ligger i uppgiftens natur. Pekarstegring valdes i ena funktionen och en whilesats valdes i den andra. I den andra valdes dessutom tilldelning via adressbas + offset. Förändringen av uppgift 4 är mer än vad som krävs för att få godkänt och visar hur väl förtrogen studenten är med C-språket. Bra!
Uppg 5 -	
Uppg 6 +	1, Kan lösas med structer men här är lösningen med en array vilket också är okej enligt uppgift. 2, Tidsuppräkningslösningen fungerar på ett godtyckligt antal sekunder 3, Mycket välskriven pseudokod
Uppg 6 -	1, I andra while(1) används kontroll mot sekunder (tid[2]) borde vara mot minuter (tid[1]) - antagligen skrivfel
Uppg 7 +	Korrekt utförd och dessutom med vissa säkerhetskontroller (tur att fgets-\n-borttagningen är bortkommenterad. \n bör nämligen vara med.)
Uppg 7 -	
Uppg 8 +	Korrekt utförd
Uppg 8 -	

BONUS

Plus-laboration	---
Alla labbar klara	G 071212

Något av följande projekt klar innan tentamen

Videobutik	---	**Altera DE2**	
Robotik		**Projdok Chat**	---

Sammanfattning	Ett skrivfel i uppgift 6 och att inte välja den optimalaste lösningen i samma uppgift kan aldrig skugga över det faktum att alla uppgifter är väl lösta på ett bra och konventionellt sätt. Bra struktur, bra val av variabel och funktionsnamn och meningsfulla kommentarer på rätt ställe kan bara ge högsta betyg. Super!
Betyg	**5**

Bedömningsunderlag för Tentamen CDT106

A

Gränsvärden för betyg

Hur man får godkänt i kursen och denna tentamen står beskrivet i kursPM, men här följer ändå en lathund som lite grovt beskriver vad som krävdes på denna tentamen. Innehåller lösningsförslagen några fel kan i första hand A-uppgifter kompenseras med bonus från laborationer och färdigställda projekt. Undantagsvis kan bonus kompensera på G-uppgifter.
3 = Klarat Uppgift 1-3 samt uppgift 4 med funktioner utan några större fel.
4 = Som för betyget 3 samt ytterligare två A-moment varav minst en måste vara förvärvad från uppgift 5 eller 8
5 = Besvarat alla uppgifter utan några större fel. Om någon A-uppgift innehåller fel kan detta kompenseras med bonus.

Bedömningsfokus
Uppg1: Grundläggande I/O, villkor _loopar, Uppg2: Definition, deklaration och anrop av funktioner, Uppg3: Analys och förståelse av nästlade loopar, Uppg4: Arrayer, funktionsanrop med arrayer, funktionsbegrepp återanvändning av kod, Uppg5: Bredd på C-språket; Uppg6: Komplexitet, problemlösningsförmåga och ev. structer; Uppg7: Filhantering, Uppg8: Förmåga att läsa funktionsdokumentation, stränghantering.
Kan man visa i andra uppgifter att man behärskar något av ovanstående så har man det självklart till godo.

Övrig information

Examinatorn har försökt att hitta såväl korrekta som felaktiga delar i lösningsförslagen och även ge tips om förbättringar. Med tanke på det stora antalet prov som ska bedömas på kort tid, kan det tyvärr bli slagsida på endera typen av kommentarer vilket inte är avsikten. Det finns självklart ingen som helst avsikt att sänka någon. Programmering ÄR tyvärr svårare än de flesta anar, och det kan därför ta tid innan man behärskar tillräckligt mycket för att kunna bli godkänd. Det är examinatorns mål att alla som vill lära sig programmera ska få tillräckligt med handledning för att kunna klara kursen. Däremot kan ingen lova hur lång tid det kan ta för att nå det målet eftersom inlärningsförmåga och tidigare erfarenheter är individuella egenskaper och inlärning är heller inte en linjär process utan går för de flesta i trappsteg som är olika höga.
Den mest komplicerade uppgiften på tentan var uppgift 6 där man skulle lägga till ett antal sekunder till en vis tid. Om man lagrar tiden i en struct tidpunkt{ int h,m,s;} och tidstillägget ligger i int s; så finns det flera sätt att lösa problemet. (a) Ett enkelt men onödigt processorintensivt är att loopa tidstillägget nedåt tills det når noll. Öka sekunder för varje varv och om det passerar 60 så ska det nollställas och minuten ökar med ett steg och vidare med timmar. (b) Istället för att öka med en sekund i taget så minskar man med 60 sekunder eller minuter i stöten om dessa variabler är större än 60, 60 och 24 och i samma veva justera de högre värdesiffrorna med ett steg. (c) Det absolut snabbaste och kortaste svaret blir dock med direkt heltalsdivision med och utan resträkning dvs; void oka(struct tidpunkt *t, int s){ t->s = (t->s+s)%60; t->m = (t->m+(s/60))%60; t->h = (t->h+(s/3600))%24; }
Om du har frågor om dina kommentarer, din tentamen eller något annat i kursen, tveka inte att kontakta Filip Sebek. Du kan självklart lämna åsikter anonymt i hans fack. Försök i så fall vara så detaljerad som möjligt.

Bedömningsunderlag för Tentamen CDT106

B

Uppg 1 +	G - Dugga (löste uppgiften trots allt ganska rätt; missade dubbla = i while-villkorsatsen)
Uppg 1 -	
Uppg 2 +	G - Dugga (löste uppgiften trots godkänd dugga)
Uppg 2 -	
Uppg 3 +	I stort sett rätt lösning (se -) vilket indikerar förståelse för programflöde, komplexa nästlade loopar och förmåga att sätta sig in i kod skriven av andra.
Uppg 3 -	Missar sista varvet i andra inre loopen vilket gör att alla rader blir en stjärna för liten
Uppg 4 +	Ganska rätt, men felaktig utmatning (se -)
Uppg 4 -	1, readVector ska läsa in värden till EN vektor och inte bägge vektorerna på en gång. 2, längden på vektorn och antalet tal i vektorn ska vara LENGTH så som den är definierad enligt uppgift 3, biggestInvector ska returnera största talet i EN vektor - inte ta in två och därefter returnera värdet på största talet i bägge vektorerna eller 0 om det största talet är lika. Programmet/funktionen svarar inte på vilken vektor som innehåller det största värdet. 4, skriver pipetecken || som absolutbelopp istället för klamrar [] - otydlig handstil samt något hoppig/otydlig indentering 5, scanf-satserna saknar &-operatorn i samband med inläsning av vektorer (genomgående i hela tentan)
Uppg 5 +	1, Valt att använda arrayadressbas plus offset istället för vanlig indexering - är dock inte pekarstegring, men är dock en variant på uppg 4 tyvärr är hanteringen felaktig (se -) 2, Använt while istället för for.
Uppg 5 -	1, Felaktig stjärna i readIn. Antingen scanf("%d",(A+i)); eller scanf("%d",&(*(A+i)); men inte scanf("%d",*(A+i)); 2, Parenteser saknas vid adressberäkningen *A+i betyder inte innehållet i A+i som i *(A+i) utan betyder (*A)+i vilket utläses innehållet i A plus i vilket är en helt annan sak. 3, Ej deklarerat och initierat variabel i i funktionen Big
Uppg 6 +	Valt att använda struct vilket är ett ganska naturligt och bra val. Använder sig av pekare till struct vid funktionsanropen på ett korrekt sätt i alla tre funktionerna Funktion andratid och visa är korrekta Pseudokoden beskriver koden korrekt men funktionen gör inte det den ska. (se -)
Uppg 6 -	1, main() ---Missat kolon i case-satserna t.ex. case 1: och missat "switch"-satsen ---Missat semikolon i funktionsanropen ---använder variabeln "v" som borde vara "val" i while-satsen ---while() saknar semikolon (och returnsats i main saknas) 2, funktionen sec(struct tid *sec) har felaktig funktionalitet; sekund-fältet bara ändras - det adderas inte till tid till tidpunkten!
Uppg 7 +	EJ LÖST UPPGIFT
Uppg 7 -	
Uppg 8 +	EJ LÖST UPPGIFT
Uppg 8 -	

BONUS

Plus-laboration	---
Alla labbar klara	G 071219

Något av följande projekt klar innan tentamen

Bedömningsunderlag för Tentamen CDT106

B

Videobutik	---	**Altera DE2**	
Robotik		**Projdok Chat**	---

Sammanfattning

Även om uppgift 4 inte är korrekt utförd så hanteras arrayer på ett tillfredställande sätt.
Sekvens-selektion och iteration hanteras korrekt och likaså det viktiga funktionsbegreppet
Tyvärr svarar du i både uppgift 4 och 6 lite på fel fråga genom att programmen gör något annat än vad de borde göra. Ska man titta på på vilken sida om gränsen du faller in så får man titta på de kvantitativa målen så uppfyller du samtliga G-avsnitt samt flera A-avsnitt (t.ex. grundläggande strukturhantering, pekarhantering, funktioner call-by-reference). Tittar man på de kvalitativa målen är det den första punkten "skriva ett program som löser ett enkelt problem) som blir aningen svårbedömt på denna tentamen. Du har dock fått godkänt på samtliga laborationer vilket visar på en viss programmeringsförmåga och jag kan därför med gott samvete ge godkänt.
(Om det beror på stress på tentan eller något annat vet jag inte, men du har blandat ihop bokstäver lite här och var och du svarar inte riktigt på frågan för det verkar som att du kan programmering bättre än du visar. I all välmening och utan anklagelse så vill jag därför att du funderar på om du inte skulle låta dig testas för dyslexi. Har du papper på det så kan du få extra tid (du satt nästan hela tiden ut) och dessutom frågorna inlästa på band (vilket kan underlätta dig att lösa uppgiften så som den är ställd). Diskutera gärna saken med mig, studievägledare Malin Åshuvud eller i första skedet med någon kompis så kan vi säkert hjälpa dig).

Betyg **3**

Gränsvärden för betyg

Hur man får godkänt i kursen och denna tentamen står beskrivet i kursPM, men här följer ändå en lathund som lite grovt beskriver vad som krävdes på denna tentamen. Innehåller lösningsförslagen några fel kan i första hand A-uppgifter kompenseras med bonus från laborationer och färdigställda projekt. Undantagsvis kan bonus kompensera på G-uppgifter.
3 = Klarat Uppgift 1-3 samt uppgift 4 med funktioner utan några större fel.
4 = Som för betyget 3 samt ytterligare två A-moment varav minst en måste vara förvärvad från uppgift 5 eller 8
5 = Besvarat alla uppgifter utan några större fel. Om någon A-uppgift innehåller fel kan detta kompenseras med bonus.

Bedömningsfokus
Uppg1: Grundläggande I/O, villkor _loopar, Uppg2: Definition, deklaration och anrop av funktioner, Uppg3: Analys och förståelse av nästlade loopar, Uppg4: Arrayer, funktionsanrop med arrayer, funktionsbegrepp återanvändning av kod, Uppg5: Bredd på C-språket; Uppg6: Komplexitet, problemlösningsförmåga och ev. structer; Uppg7: Filhantering, Uppg8: Förmåga att läsa funktionsdokumentation, stränghantering.
Kan man visa i andra uppgifter att man behärskar något av ovanstående så har man det självklart till godo.

Övrig information

Examinatorn har försökt att hitta såväl korrekta som felaktiga delar i lösningsförslagen och även ge tips om förbättringar. Med tanke på det stora antalet prov som ska bedömas på kort tid, kan det tyvärr bli slagsida på endera typen av kommentarer vilket inte är avsikten. Det finns självklart ingen som helst avsikt att sänka någon. Programmering ÄR tyvärr svårare än de flesta anar, och det kan därför ta tid innan man behärskar tillräckligt mycket för att kunna bli godkänd. Det är examinatorns mål att alla som vill lära sig programmera ska få tillräckligt med handledning för att kunna klara kursen. Däremot kan ingen lova hur lång tid det kan ta för att nå det målet eftersom inlärningsförmåga och tidigare erfarenheter är individuella egenskaper och inlärning är heller inte en linjär process utan går för de flesta i trappsteg som är olika höga.
Den mest komplicerade uppgiften på tentan var uppgift 6 där man skulle lägga till ett antal sekunder till en vis tid. Om man lagrar tiden i en struct tidpunkt{ int h,m,s;} och tidstillägget ligger i int s; så finns det flera sätt att lösa problemet. (a) Ett enkelt men onödigt processorintensivt är att loopa tidstillägget nedåt tills det når noll. Öka sekunder för varje varv och om det passerar 60 så ska det nollställas och minuten ökar med ett steg och vidare med timmar. (b) Istället för att öka med en sekund i taget så minskar man med 60 sekunder eller minuter i stöten om dessa variabler är större än 60, 60 och 24 och i samma veva justera de högre värdesiffrorna med ett steg. (c) Det absolut snabbaste och kortaste svaret blir dock med direkt heltalsdivision med och utan resträkning dvs; void oka(struct tidpunkt *t, int s){ t->s = (t->s+s)%60; t->m = (t->m+(s/60))%60; t->h = (t->h+(s/3600))%24; }
Om du har frågor om dina kommentarer, din tentamen eller något annat i kursen, tveka inte att kontakta Filip Sebek. Du kan självklart lämna åsikter anonymt i hans fack. Försök i så fall vara så detaljerad som möjligt.

Bedömningsunderlag för Tentamen CDT106

C

Uppg 1 +	G - Dugga
Uppg 1 -	
Uppg 2 +	G - Dugga
Uppg 2 -	
Uppg 3 +	Korrekt utskrift vilket indikerar förståelse för programflöde, komplexa nästlade loopar och förmåga att sätta sig in i kod skriven av andra.
Uppg 3 -	
Uppg 4 +	Konventionell lösning med indexering, men visar även upp hur man kan accessa vektorer med hjälp av pekarbas + offset
Uppg 4 -	Saknar satsavslutande semikolon vid funktionsdeklarationerna.
Uppg 5 +	Intressant att välja kapsla in datat med metadata i en struct
Uppg 5 -	I enternumbers loopar du inte över v[] så du fyller bara på vektor 0 med LENGTH antal värden - synd för jag gillar idén!
Uppg 6 +	1, Använder sig av struct för att lagra tidpunkten. 2, Säkrar upp inmatningen så att endast korrekta tider matas in av användaren 3, Kan addera till godtyckligt antal sekunder i addsecondsfunktionen
Uppg 6 -	1, switch-case-konstruktionen lite fel syntax-mässig. Ska vara case 1: (KOLON) och inga klamrar 2, I funktionerna där du tar in structen som en pekare tilldelar du fälten med en onödig stjärna. Antingen så ska du göra oldtime ->hour =dummy; eller (*oldtime).hour=dummy; 3, felaktig return i void-funktion på printtime()
Uppg 7 +	Även om programmet inte fungerar som det ska så är det i princip tankemässigt korrekt. (se -)
Uppg 7 -	1, Felaktig * i fopen(*filename,"r"); ta bort 2, Genom att while-sätta thefile så kommer funktionen fastna i en oändlig loop eftersom filen inte finns - okej användaren kan ju få för sig att kopiera in den och då rasslar programmet vidare… 3, "w"-läge raderar befintlig fil och därför kan du inte spola fram i den. Du borde valt "a"-läge för att lägga till raden. Spola fram heter för övrigt fseek(thefile,0,SEEK_END);
Uppg 8 +	EJ LÖST.
Uppg 8 -	

BONUS

Plus-laboration	---
Alla labbar klara	G 071217

Något av följande projekt klar innan tentamen

Videobutik	---	**Altera DE2**	
Robotik		**Projdok Chat**	---

Sammanfattning	På det hela taget en väl genomförd tentamen. Sista uppgiften visserligen inte löst pga tidsnöd. Resultatet är en solklar 4:a hade du löst +-uppgiften eller projektet innan tentan eller löst sista uppgiften på tentan så hade det blivit en 5:a.

Bedömningsunderlag för Tentamen CDT106

C

Betyg	**4**

Gränsvärden för betyg

Hur man får godkänt i kursen och denna tentamen står beskrivet i kursPM, men här följer ändå en lathund som lite grovt beskriver vad som krävdes på denna tentamen. Innehåller lösningsförslagen några fel kan i första hand A-uppgifter kompenseras med bonus från laborationer och färdigställda projekt. Undantagsvis kan bonus kompensera på G-uppgifter.
3 = Klarat Uppgift 1-3 samt uppgift 4 med funktioner utan några större fel.
4 = Som för betyget 3 samt ytterligare två A-moment varav minst en måste vara förvärvad från uppgift 5 eller 8
5 = Besvarat alla uppgifter utan några större fel. Om någon A-uppgift innehåller fel kan detta kompenseras med bonus.

Bedömningsfokus
Uppg1: Grundläggande I/O, villkor loopar, Uppg2: Definition, deklaration och anrop av funktioner, Uppg3: Analys och förståelse av nästlade loopar, Uppg4: Arrayer, funktionsanrop med arrayer, funktionsbegrepp återanvändning av kod, Uppg5: Bredd på C-språket; Uppg6: Komplexitet, problemlösningsförmåga och ev. structer; Uppg7: Filhantering, Uppg8: Förmåga att läsa funktionsdokumentation, stränghantering.
Kan man visa i andra uppgifter att man behärskar något av ovanstående så har man det självklart till godo.

Övrig information

Examinatorn har försökt att hitta såväl korrekta som felaktiga delar i lösningsförslagen och även ge tips om förbättringar. Med tanke på det stora antalet prov som ska bedömas på kort tid, kan det tyvärr bli slagsida på endera typen av kommentarer vilket inte är avsikten. Det finns självklart ingen som helst avsikt att sänka någon. Programmering ÄR tyvärr svårare än de flesta anar, och det kan därför ta tid innan man behärskar tillräckligt mycket för att kunna bli godkänd. Det är examinatorns mål att alla som vill lära sig programmera ska få tillräckligt med handledning för att kunna klara kursen. Däremot kan ingen lova hur lång tid det kan ta för att nå det målet eftersom inlärningsförmåga och tidigare erfarenheter är individuella egenskaper och inlärning är heller inte en linjär process utan går för de flesta i trappsteg som är olika höga.
Den mest komplicerade uppgiften på tentan var uppgift 6 där man skulle lägga till ett antal sekunder till en vis tid. Om man lagrar tiden i en struct tidpunkt{ int h,m,s;} och tidstillägget ligger i int s; så finns det flera sätt att lösa problemet. (a) Ett enkelt men onödigt processorintensivt är att loopa tidstillägget nedåt tills det når noll. Öka sekunder för varje varv och om det passerar 60 så ska det nollställas och minuten ökar med ett steg och vidare med timmar. (b) Istället för att öka med en sekund i taget så minskar man med 60 sekunder eller minuter i stöten om dessa variabler är större än 60, 60 och 24 och i samma veva justera de högre värdesiffrorna med ett steg. (c) Det absolut snabbaste och kortaste svaret blir dock med direkt heltalsdivision med och utan resträkning dvs; void oka(struct tidpunkt *t, int s){ t->s = (t->s+s)%60; t->m = (t->m+(s/60))%60; t->h = (t->h+(s/3600))%24; }
Om du har frågor om dina kommentarer, din tentamen eller något annat i kursen, tveka inte att kontakta Filip Sebek. Du kan självklart lämna åsikter anonymt i hans fack. Försök i så fall vara så detaljerad som möjligt.

Bedömningsunderlag för Tentamen CDT106

D

Uppg 1 +	Algoritmiskt ganska rätt med några syntaktiska missar
Uppg 1 -	main() saknas x upphöjt till två går inte att skriva så matematiskt i C; antingen pow(x,2) eller x*x
Uppg 2 +	korrekt funktionsdeklaration, anrop och definition
Uppg 2 -	main() saknas
Uppg 3 +	Endast första raden med 7 * korrekt (bara skrivit en rad av 7)
Uppg 3 -	Går ej att bedöma hur mycket som är greppat - för lite underlag
Uppg 4 +	I stort ett korrekt program med korrekt vektorhantering och funktionsanrop
Uppg 4 -	1, main() saknas 2, felaktig retur-sats. Ska utelämnas helt (void-funktion) 3, funktionen biggestinvector utgår från att största talet är minst 0. Säkrare/mer rätt att initiera till biggest=a[0] och börja loopen med for(i=1 . . .) 4, vad används inargumentet limit till i biggestinvector?
Uppg 5 +	EJ LÖST UPPGIFT
Uppg 5 -	
Uppg 6 +	Endast pseudokod inlämnad. Principiellt rätt, kan dock inte ge A-poäng
Uppg 6 -	C-kod saknas. Sparas i tre separata int-variabler
Uppg 7 +	EJ LÖST UPPGIFT
Uppg 7 -	
Uppg 8 +	EJ LÖST UPPGIFT
Uppg 8 -	

BONUS

Plus-laboration	---
Alla labbar klara	---

Något av följande projekt klar innan tentamen

Videobutik	---	**Altera DE2**	
Robotik		**Projdok Chat**	---

Sammanfattning	Flera mindre syntaktiska fel som tyder på för lite praktisk programmeirng i just språket C. Eftersom G-uppgiften 3 är fel och uppg 4 även den innehåller vissa fel och endast ett A-moment är uppfyllt (uppg 4)
Betyg	**U**

Bedömningsunderlag för Tentamen CDT106

D

Gränsvärden för betyg

Hur man får godkänt i kursen och denna tentamen står beskrivet i kursPM, men här följer ändå en lathund som lite grovt beskriver vad som krävdes på denna tentamen. Innehåller lösningsförslagen några fel kan i första hand A-uppgifter kompenseras med bonus från laborationer och färdigställda projekt. Undantagsvis kan bonus kompensera på G-uppgifter.
3 = Klarat Uppgift 1-3 samt uppgift 4 med funktioner utan några större fel.
4 = Som för betyget 3 samt ytterligare två A-moment varav minst en måste vara förvärvad från uppgift 5 eller 8
5 = Besvarat alla uppgifter utan några större fel. Om någon A-uppgift innehåller fel kan detta kompenseras med bonus.

Bedömningsfokus
Uppg1: Grundläggande I/O, villkor _loopar, Uppg2: Definition, deklaration och anrop av funktioner, Uppg3: Analys och förståelse av nästlade loopar, Uppg4: Arrayer, funktionsanrop med arrayer, funktionsbegrepp återanvändning av kod, Uppg5: Bredd på C-språket; Uppg6: Komplexitet, problemlösningsförmåga och ev. structer; Uppg7: Filhantering, Uppg8: Förmåga att läsa funktionsdokumentation, stränghantering.
Kan man visa i andra uppgifter att man behärskar något av ovanstående så har man det självklart till godo.

Övrig information

Examinatorn har försökt att hitta såväl korrekta som felaktiga delar i lösningsförslagen och även ge tips om förbättringar. Med tanke på det stora antalet prov som ska bedömas på kort tid, kan det tyvärr bli slagsida på endera typen av kommentarer vilket inte är avsikten. Det finns självklart ingen som helst avsikt att sänka någon. Programmering ÄR tyvärr svårare än de flesta anar, och det kan därför ta tid innan man behärskar tillräckligt mycket för att kunna bli godkänd. Det är examinatorns mål att alla som vill lära sig programmera ska få tillräckligt med handledning för att kunna klara kursen. Däremot kan ingen lova hur lång tid det kan ta för att nå det målet eftersom inlärningsförmåga och tidigare erfarenheter är individuella egenskaper och inlärning är heller inte en linjär process utan går för de flesta i trappsteg som är olika höga.
Den mest komplicerade uppgiften på tentan var uppgift 6 där man skulle lägga till ett antal sekunder till en vis tid. Om man lagrar tiden i en struct tidpunkt{ int h,m,s;} och tidstillägget ligger i int s; så finns det flera sätt att lösa problemet. (a) Ett enkelt men onödigt processorintensivt är att loopa tidstillägget nedåt tills det når noll. Öka sekunder för varje varv och om det passerar 60 så ska det nollställas och minuten ökar med ett steg och vidare med timmar. (b) Istället för att öka med en sekund i taget så minskar man med 60 sekunder eller minuter i stöten om dessa variabler är större än 60, 60 och 24 och i samma veva justera de högre värdesiffrorna med ett steg. (c) Det absolut snabbaste och kortaste svaret blir dock med direkt heltalsdivision med och utan resträkning dvs; void oka(struct tidpunkt *t, int s){ t->s = (t->s+s)%60; t->m = (t->m+(s/60))%60; t->h = (t->h+(s/3600))%24; }
Om du har frågor om dina kommentarer, din tentamen eller något annat i kursen, tveka inte att kontakta Filip Sebek. Du kan självklart lämna åsikter anonymt i hans fack. Försök i så fall vara så detaljerad som möjligt.

Bedömningsunderlag för Tentamen CDT106

E

Uppg 1 +	G - Dugga
Uppg 1 -	
Uppg 2 +	Korrekt program utan fel korrekt funktionsdeklaration, anrop och definition
Uppg 2 -	
Uppg 3 +	.
Uppg 3 -	felaktig analys av programmet. Beskrivningen tyder på att begrepp som inre och yttre loop inte är fullt greppat.
Uppg 4 +	Med undantag för syntaxen för parameteröverföring av array så är programmet korrekt utfört med deklaration och användandet av arrayer med indexering
Uppg 4 -	1, felaktigt syntax vid funktionsanrop med vektor som inparameter readVector(array[]); borde vara readVector(array); i övrigt korrekt funktion
Uppg 5 +	Istället för indexering har accessen skett med adressbas plus offset (vilket inte är samma sak som pekarstegring) men dock en variation while använt istället för for
Uppg 5 -	felaktigt användade av scanf som vill ha in en adress. Din lösning är *(array+i) vilket egentligen borde vara &(*(array+i)) eller enklare (array+i) dvs utan innehållsoperatorn *
Uppg 6 +	valt att lagra tiden som en struct. Delvis korrekt lösning Välskriven pseudokod
Uppg 6 -	1, (main ganska stor - borde delas upp i fler funktioner) 2, variablen tid är inte deklarerad i main() däremot är tid definierad som en structtyp 3, addsec har vissa buggar. Om tiden är 10:50:55 och vi lägger till 190 sekunder så blir första delresultatet först 10:51:185 och därefter slutresultatet 10:52:125 vilket inte är korrekt...
Uppg 7 +	Nästan helt korrekt lösning
Uppg 7 -	fclose saknas De flesta operativsystem hindrar öppna filer från att bli uppdaterade, man bör därför först öppna filen, läsa raden, stängda den och först därefter öppna den för uppdatering
Uppg 8 +	Till stora delar korrekt sånär som pekarhantering
Uppg 8 -	strlen och strcat ska ha en pekare till strängen. Att sätta *str betyder att första tecknet skickas in som inargument. Följaktligen borde det vara strcat(str,kort);

BONUS

Plus-laboration	---
Alla labbar klara	G 071220

Något av följande projekt klar innan tentamen

Videobutik	---	**Altera DE2**	
Robotik		**Projdok Chat**	8.00

Sammanfattning	Att försöka lösa alla uppgifter på ett seriöst sätt med alla de konstruktioner som finns i C vittnar om stor bredd i kunskaper. De flesta uppgifterna är ganska rätt lösta, men innehåller samtidigt en hel del missar. I synnerhet verkar hanteringen av pekare inte vara fullt greppad vilket vittnas i uppgift 4,5 och 8. Uppgift 3 är fel löst och

Bedömningsunderlag för Tentamen CDT106

E

	Uppgift 6 är inte så praktiskt löst och därför kan det även om en bonus från färdiga labbar finns inte bli fråga om det högsta betyget men väl en knapp 4:a. Visserligen är G-uppgift 3 felaktigt löst men det finns tillräckligt med underlag att kunna bedöma att den som skrivit tentan har kompetensen att kunna programmera enligt de övergripande målen med kursen. Med lite mer programmeringsrutin så skulle betyget 5 kunna erhållas.
Betyg	**4**

Gränsvärden för betyg

Hur man får godkänt i kursen och denna tentamen står beskrivet i kursPM, men här följer ändå en lathund som lite grovt beskriver vad som krävdes på denna tentamen. Innehåller lösningsförslagen några fel kan i första hand A-uppgifter kompenseras med bonus från laborationer och färdigställda projekt. Undantagsvis kan bonus kompensera på G-uppgifter.
3 = Klarat Uppgift 1-3 samt uppgift 4 med funktioner utan några större fel.
4 = Som för betyget 3 samt ytterligare två A-moment varav minst en måste vara förvärvad från uppgift 5 eller 8
5 = Besvarat alla uppgifter utan några större fel. Om någon A-uppgift innehåller fel kan detta kompenseras med bonus.

Bedömningsfokus
Uppg1: Grundläggande I/O, villkor _loopar, Uppg2: Definition, deklaration och anrop av funktioner, Uppg3: Analys och förståelse av nästlade loopar, Uppg4: Arrayer, funktionsanrop med arrayer, funktionsbegrepp återanvändning av kod, Uppg5: Bredd på C-språket; Uppg6: Komplexitet, problemlösningsförmåga och ev. structer; Uppg7: Filhantering, Uppg8: Förmåga att läsa funktionsdokumentation, stränghantering.
Kan man visa i andra uppgifter att man behärskar något av ovanstående så har man det självklart till godo.

Övrig information

Examinatorn har försökt att hitta såväl korrekta som felaktiga delar i lösningsförslagen och även ge tips om förbättringar. Med tanke på det stora antalet prov som ska bedömas på kort tid, kan det tyvärr bli slagsida på endera typen av kommentarer vilket inte är avsikten. Det finns självklart ingen som helst avsikt att sänka någon. Programmering ÄR tyvärr svårare än de flesta anar, och det kan därför ta tid innan man behärskar tillräckligt mycket för att kunna bli godkänd. Det är examinatorns mål att alla som vill lära sig programmera ska få tillräckligt med handledning för att kunna klara kursen. Däremot kan ingen lova hur lång tid det kan ta för att nå det målet eftersom inlärningsförmåga och tidigare erfarenheter är individuella egenskaper och inlärning är heller inte en linjär process utan går för de flesta i trappsteg som är olika höga.
Den mest komplicerade uppgiften på tentan var uppgift 6 där man skulle lägga till ett antal sekunder till en vis tid. Om man lagrar tiden i en struct tidpunkt{ int h,m,s;} och tidstillägget ligger i int s; så finns det flera sätt att lösa problemet. (a) Ett enkelt men onödigt processorintensivt är att loopa tidstillägget nedåt tills det når noll. Öka sekunder för varje varv och om det passerar 60 så ska det nollställas och minuten ökar med ett steg och vidare med timmar. (b) Istället för att öka med en sekund i taget så minskar man med 60 sekunder eller minuter i stöten om dessa variabler är större än 60, 60 och 24 och i samma veva justera de högre värdesiffrorna med ett steg. (c) Det absolut snabbaste och kortaste svaret blir dock med direkt heltalsdivision med och utan resträkning dvs; void oka(struct tidpunkt *t, int s){ t->s = (t->s+s)%60; t->m = (t->m+(s/60))%60; t->h = (t->h+(s/3600))%24; }
Om du har frågor om dina kommentarer, din tentamen eller något annat i kursen, tveka inte att kontakta Filip Sebek. Du kan självklart lämna åsikter anonymt i hans fack. Försök i så fall vara så detaljerad som möjligt.

Bedömningsunderlag för Tentamen CDT106

F

Uppg 1 +	Påbörjat men inte avslutat lösningsförslag
Uppg 1 -	1, Programmet saknar inläsningsfunktioner med scanf() eller gets. Printf() skriver ju bara ut 2, någon while/loop finns inte
Uppg 2 +	funktionsbegreppet är till stora delar korrekt korrekt funktionsnarop och i princip rätt funktionsdefinition
Uppg 2 -	funktionsdeklaration saknas felaktigt semikolon på funktionshuvudet i funktionsdefinitionen returnerar int när det borde vara float
Uppg 3 +	Förstått att det är flera rader som ska skrivas ut
Uppg 3 -	skriver ut en rad för mycket och har inte greppet vad som ska skrivas ut i varje rad med de nästalde looparna
Uppg 4 +	viss korrekt I/O men löst på ett felaktigt sätt enligt specifikationen i uppgiften (se -)
Uppg 4 -	1, Returnerar två vektorer från readvector vilket är omöjligt och onödigt med tanke på att vektorernas referenser/adresser skickas in till funktionen 2, readvector ska endast läsa in EN vektor - du skickar två i funktionsanropet till readvector skickar du ett element i vektorn - inte hela vektorn 3, biggestvector hittar inte största elementet i vektorn - den returnerar heller inget värde utan skriver ut "svaret" på skärmen.
Uppg 5 +	EJ LÖST UPPGIFT
Uppg 5 -	
Uppg 6 +	Menybaserat program enligt specifikation i uppgiften - dock kan man bara göra ett val en gång - menyn är inte i en loop
Uppg 6 -	Tidpunkten är lagrad som ett heltal initierad som en textsträng vilket är omöjligt. Programmet saknar scanf-satser och vissa printfsatser används felaktigt tidpunkten är inte separerad i timmar, minuter och skender enligt uppgiftsformuleringen felaktigt funktionalitet i seccalc
Uppg 7 +	flera principiellt korrekta delar i lösningen, men ändå många fel
Uppg 7 -	1, fopen görs inte med den generella variablen filename utan på den hårdkodade "filename.txt". Oläsbar metod A? 2, Om "A" används så kommer tillägg göras direkt - programmet kommer inte att kunna läsa från filen 3, Felaktig *. Ska vara fpekare=fopen(filename,"A"); 4, fgets() arbetar inte mot filnamn utan mot filpekare exvis fgets(dummy,88,fpekare);
Uppg 8 +	EJ LÖST UPPGIFT
Uppg 8 -	

BONUS

Plus-laboration	---
Alla labbar klara	---

Något av följande projekt klar innan tentamen

Videobutik	---	**Altera DE2**	
Robotik		**Projdok Chat**	---

Sammanfattning	Brister i I/O-hantering. Sekvens och enklare selektion fungerar, loopar/iterationer mera tveksamt. Visar prov

Bedömningsunderlag för Tentamen CDT106

F

	på små kunskaper här och var men har inte greppat något större eller viktigare avsnitt helt. Du behöver förmodligen mer tid och öva mera/ göra alla alborationer innan programmeringen börjar gå din väg.
Betyg	**U**

Gränsvärden för betyg

Hur man får godkänt i kursen och denna tentamen står beskrivet i kursPM, men här följer ändå en lathund som lite grovt beskriver vad som krävdes på denna tentamen. Innehåller lösningsförslagen några fel kan i första hand A-uppgifter kompenseras med bonus från laborationer och färdigställda projekt. Undantagsvis kan bonus kompensera på G-uppgifter.
3 = Klarat Uppgift 1-3 samt uppgift 4 med funktioner utan några större fel.
4 = Som för betyget 3 samt ytterligare två A-moment varav minst en måste vara förvärvad från uppgift 5 eller 8
5 = Besvarat alla uppgifter utan några större fel. Om någon A-uppgift innehåller fel kan detta kompenseras med bonus.

Bedömningsfokus
Uppg1: Grundläggande I/O, villkor _loopar, Uppg2: Definition, deklaration och anrop av funktioner, Uppg3: Analys och förståelse av nästlade loopar, Uppg4: Arrayer, funktionsanrop med arrayer, funktionsbegrepp återanvändning av kod, Uppg5: Bredd på C-språket; Uppg6: Komplexitet, problemlösningsförmåga och ev. structer; Uppg7: Filhantering, Uppg8: Förmåga att läsa funktionsdokumentation, stränghantering.
Kan man visa i andra uppgifter att man behärskar något av ovanstående så har man det självklart till godo.

Övrig information

Examinatorn har försökt att hitta såväl korrekta som felaktiga delar i lösningsförslagen och även ge tips om förbättringar. Med tanke på det stora antalet prov som ska bedömas på kort tid, kan det tyvärr bli slagsida på endera typen av kommentarer vilket inte är avsikten. Det finns självklart ingen som helst avsikt att sänka någon. Programmering ÄR tyvärr svårare än de flesta anar, och det kan därför ta tid innan man behärskar tillräckligt mycket för att kunna bli godkänd. Det är examinatorns mål att alla som vill lära sig programmera ska få tillräckligt med handledning för att kunna klara kursen. Däremot kan ingen lova hur lång tid det kan ta för att nå det målet eftersom inlärningsförmåga och tidigare erfarenheter är individuella egenskaper och inlärning är heller inte en linjär process utan går för de flesta i trappsteg som är olika höga.
Den mest komplicerade uppgiften på tentan var uppgift 6 där man skulle lägga till ett antal sekunder till en vis tid. Om man lagrar tiden i en struct tidpunkt{ int h,m,s;} och tidstillägget ligger i int s; så finns det flera sätt att lösa problemet. (a) Ett enkelt men onödigt processorintensivt är att loopa tidstillägget nedåt tills det når noll. Öka sekunder för varje varv och om det passerar 60 så ska det nollställas och minuten ökar med ett steg och vidare med timmar. (b) Istället för att öka med en sekund i taget så minskar man med 60 sekunder eller minuter i stöten om dessa variabler är större än 60, 60 och 24 och i samma veva justera de högre värdesiffrorna med ett steg. (c) Det absolut snabbaste och kortaste svaret blir dock med direkt heltalsdivision med och utan resträkning dvs; void oka(struct tidpunkt *t, int s){ t->s = (t->s+s)%60; t->m = (t->m+(s/60))%60; t->h = (t->h+(s/3600))%24; }
Om du har frågor om dina kommentarer, din tentamen eller något annat i kursen, tveka inte att kontakta Filip Sebek. Du kan självklart lämna åsikter anonymt i hans fack. Försök i så fall vara så detaljerad som möjligt.

Bedömningsunderlag för Tentamen CDT106

G

Uppg 1 +	G - Dugga
Uppg 1 -	
Uppg 2 +	G - Dugga
Uppg 2 -	
Uppg 3 +	.
Uppg 3 -	Felaktig utskrift. Greppat att det ska vara ett antal rader, men istället för 7 rader så blev det 6 helt felaktiga. Går ej att bedöma hur mycket som är greppat - för lite underlag i lösningsförslaget
Uppg 4 +	Korrekta deklarationer och tilldelningar av vektorer. Något ofullständigt program korrekt parameteröverföring av vektor(er) till funktion
Uppg 4 -	1, felaktig * i scanf i readvector; borde vara scanf("%d",arrPek1); eller scanf("%d",&(*arrPek1)); 2, Funktionen readvector gör visserligen rätt saker i sammanhanget men inte enligt uppgiften. Den ska läsa in EN vektor inte två 3, klamrar kring funktionsdefinitionen på biggestvector saknas 4, Varför söks max i de två vektorerna i main (överst på blad 2), när det redan är hittat i funktionen som anropas på blad 1 ? Max används visserligen inte men . . .
Uppg 5 +	EJ LÖST UPPGIFT
Uppg 5 -	
Uppg 6 +	EJ LÖST UPPGIFT
Uppg 6 -	
Uppg 7 +	principiellt korrekt, men med några logiska och syntaktiska missar
Uppg 7 -	1, fopen görs inte med den generella variablen filename utan på den hårdkodade "filename.txt". 2, omotiverad while-satser - däremot är den inkapslade gets-och puts-satsen korrekta 3, öppnar man i läget "a" så behöver man inte spola fram till EOF - man hamnar där automatiskt.
Uppg 8 +	EJ LÖST UPPGIFT
Uppg 8 -	

BONUS

Plus-laboration	---
Alla labbar klara	---

Något av följande projekt klar innan tentamen

Videobutik	---	**Altera DE2**	
Robotik		**Projdok Chat**	---

Sammanfattning	Grunderna i variabelhantering, loopar och selektion finns där. Brister i lite i problemlösning och skulle behöva något avancerat momnet till En god bit på väg mot målet, men en liten bit kvar.
Betyg	**U**

Bedömningsunderlag för Tentamen CDT106

G

Gränsvärden för betyg

Hur man får godkänt i kursen och denna tentamen står beskrivet i kursPM, men här följer ändå en lathund som lite grovt beskriver vad som krävdes på denna tentamen. Innehåller lösningsförslagen några fel kan i första hand A-uppgifter kompenseras med bonus från laborationer och färdigställda projekt. Undantagsvis kan bonus kompensera på G-uppgifter.
3 = Klarat Uppgift 1-3 samt uppgift 4 med funktioner utan några större fel.
4 = Som för betyget 3 samt ytterligare två A-moment varav minst en måste vara förvärvad från uppgift 5 eller 8
5 = Besvarat alla uppgifter utan några större fel. Om någon A-uppgift innehåller fel kan detta kompenseras med bonus.

Bedömningsfokus
Uppg1: Grundläggande I/O, villkor _loopar, Uppg2: Definition, deklaration och anrop av funktioner, Uppg3: Analys och förståelse av nästlade loopar, Uppg4: Arrayer, funktionsanrop med arrayer, funktionsbegrepp återanvändning av kod, Uppg5: Bredd på C-språket; Uppg6: Komplexitet, problemlösningsförmåga och ev. structer; Uppg7: Filhantering, Uppg8: Förmåga att läsa funktionsdokumentation, stränghantering.
Kan man visa i andra uppgifter att man behärskar något av ovanstående så har man det självklart till godo.

Övrig information

Examinatorn har försökt att hitta såväl korrekta som felaktiga delar i lösningsförslagen och även ge tips om förbättringar. Med tanke på det stora antalet prov som ska bedömas på kort tid, kan det tyvärr bli slagsida på endera typen av kommentarer vilket inte är avsikten. Det finns självklart ingen som helst avsikt att sänka någon. Programmering ÄR tyvärr svårare än de flesta anar, och det kan därför ta tid innan man behärskar tillräckligt mycket för att kunna bli godkänd. Det är examinatorns mål att alla som vill lära sig programmera ska få tillräckligt med handledning för att kunna klara kursen. Däremot kan ingen lova hur lång tid det kan ta för att nå det målet eftersom inlärningsförmåga och tidigare erfarenheter är individuella egenskaper och inlärning är heller inte en linjär process utan går för de flesta i trappsteg som är olika höga.
Den mest komplicerade uppgiften på tentan var uppgift 6 där man skulle lägga till ett antal sekunder till en vis tid. Om man lagrar tiden i en struct tidpunkt{ int h,m,s;} och tidstillägget ligger i int s; så finns det flera sätt att lösa problemet. (a) Ett enkelt men onödigt processorintensivt är att loopa tidstillägget nedåt tills det når noll. Öka sekunder för varje varv och om det passerar 60 så ska det nollställas och minuten ökar med ett steg och vidare med timmar. (b) Istället för att öka med en sekund i taget så minskar man med 60 sekunder eller minuter i stöten om dessa variabler är större än 60, 60 och 24 och i samma veva justera de högre värdesiffrorna med ett steg. (c) Det absolut snabbaste och kortaste svaret blir dock med direkt heltalsdivision med och utan resträkning dvs; void oka(struct tidpunkt *t, int s){ t->s = (t->s+s)%60; t->m = (t->m+(s/60))%60; t->h = (t->h+(s/3600))%24; }
Om du har frågor om dina kommentarer, din tentamen eller något annat i kursen, tveka inte att kontakta Filip Sebek. Du kan självklart lämna åsikter anonymt i hans fack. Försök i så fall vara så detaljerad som möjligt.

Bedömningsunderlag för Tentamen CDT106

H

Uppg 1 +	G - Dugga
Uppg 1 -	
Uppg 2 +	G - Dugga
Uppg 2 -	
Uppg 3 +	Felaktig utskrift. Greppat att det ska vara 7 rader, och där den första raden bara har * och att 4-6 inte har någon * vilket är rätt men lite väl ospecat. Går ej att bedöma hur resonemangen varit - för lite underlag i lösningsförslaget
Uppg 3 -	
Uppg 4 +	Programmet är i sin principella uppbyggnad korrekt men kommer inte att fungera enligt specifikationen.
Uppg 4 -	1, readVector är en voidfunktion som inte returnerar något (!) Jobbar med två vektorer trots att det stod i uppgiften att den skulle hantera EN vektor (måste anropas två gånger) 2, felaktig tilldelning av hel vektor i funktionen biggestinvector till comparison. 3, vektorn får inte ha enbart negativa värden då max är initierad till 0. Säkrare/mer rätt att initiera till max=x[0] och börja loopen med for(i=1 . . .)
Uppg 5 +	EJ LÖST UPPGIFT
Uppg 5 -	
Uppg 6 +	1, Lagrar tidpunkten med en struct 2, Konceptet att översätta ett flyttal som inehåller minut,sekund går nog att implementera med heltalsomvandlingar etc. men i lösningsförslagset som presenteras finns många missar vid informationsöverföringarna varför det presenterade implementationsförslaget inte fungerar då det inte lägger till tidstillägget till den aktuella tiden alls. 3, Väl beskriven pseduokod - tyvärr med är lösningsförslaget behäftat med fel.
Uppg 6 -	1, lokalt definierad struct - måste vara globalt definierad (obs! inte deklarerad) för att programmens funktionsanrop ska fungera 2, switch saknar klamrar 3, programet borde modulariseras i fler funktioner 4, funktionsdeklaration saknas 5, type mismatch på samtliga funktionsargument i addseconds. Borde vara struct tidpunkt addseconds(tidpunkt structen, int sek) { 6, addseconds tar inte hänsyn till sekund, minut och timmesövergångar. Din tidpunkt kan bli 10h62m137s
Uppg 7 +	EJ LÖST UPPGIFT
Uppg 7 -	
Uppg 8 +	1, strcat korrekt använd - skelettet runt om innehåller dock många fel
Uppg 8 -	1, if(strlen(*str<21)) måste skrivas om till if(strlen(str)<21 2, kopiering av sträng (behöver inte göras i denna uppgift) görs inte med = utan med strcpy() 3, return(mening); i en voidfunktion är fel, return av en vektor går inte, return av vektor är onödig om manipuleringen sker på variablen str direkt

BONUS

Plus-laboration	---
Alla labbar klara	---

Något av följande projekt klar innan tentamen

Videobutik	---	**Altera DE2**	

Bedömningsunderlag för Tentamen CDT106

H

Robotik		**Projdok Chat**	---

Sammanfattning	De grundläggande delarna i I/O, selektion, iteration, och variabelhanteringär avklarade. Problemlösningsförmågan är godkänd. Däremot finns brister i hantering av de grundläggande delarna i arrayer men även funktioner (och kombinationen därav). Dessa konstruktioner har lika ofta fel som de är rätt vilket tyder på en viss osäkerhet. En mycket god bit på väg mot målet, men ändå en bit kvar för att kunna få godkänt.
Betyg	**U**

Gränsvärden för betyg

Hur man får godkänt i kursen och denna tentamen står beskrivet i kursPM, men här följer ändå en lathund som lite grovt beskriver vad som krävdes på denna tentamen. Innehåller lösningsförslagen några fel kan i första hand A-uppgifter kompenseras med bonus från laborationer och färdigställda projekt. Undantagsvis kan bonus kompensera på G-uppgifter.
3 = Klarat Uppgift 1-3 samt uppgift 4 med funktioner utan några större fel.
4 = Som för betyget 3 samt ytterligare två A-moment varav minst en måste vara förvärvad från uppgift 5 eller 8
5 = Besvarat alla uppgifter utan några större fel. Om någon A-uppgift innehåller fel kan detta kompenseras med bonus.

Bedömningsfokus
Uppg1: Grundläggande I/O, villkor loopar, Uppg2: Definition, deklaration och anrop av funktioner, Uppg3: Analys och förståelse av nästlade loopar, Uppg4: Arrayer, funktionsanrop med arrayer, funktionsbegrepp återanvändning av kod, Uppg5: Bredd på C-språket; Uppg6: Komplexitet, problemlösningsförmåga och ev. structer; Uppg7: Filhantering, Uppg8: Förmåga att läsa funktionsdokumentation, stränghantering.
Kan man visa i andra uppgifter att man behärskar något av ovanstående så har man det självklart till godo.

Övrig information

Examinatorn har försökt att hitta såväl korrekta som felaktiga delar i lösningsförslagen och även ge tips om förbättringar. Med tanke på det stora antalet prov som ska bedömas på kort tid, kan det tyvärr bli slagsida på endera typen av kommentarer vilket inte är avsikten. Det finns självklart ingen som helst avsikt att sänka någon. Programmering ÄR tyvärr svårare än de flesta anar, och det kan därför ta tid innan man behärskar tillräckligt mycket för att kunna bli godkänd. Det är examinatorns mål att alla som vill lära sig programmera ska få tillräckligt med handledning för att kunna klara kursen. Däremot kan ingen lova hur lång tid det kan ta för att nå det målet eftersom inlärningsförmåga och tidigare erfarenheter är individuella egenskaper och inlärning är heller inte en linjär process utan går för de flesta i trappsteg som är olika höga.
Den mest komplicerade uppgiften på tentan var uppgift 6 där man skulle lägga till ett antal sekunder till en vis tid. Om man lagrar tiden i en struct tidpunkt{ int h,m,s;} och tidstillägget ligger i int s; så finns det flera sätt att lösa problemet. (a) Ett enkelt men onödigt processorintensivt är att loopa tidstillägget nedåt tills det når noll. Öka sekunder för varje varv och om det passerar 60 så ska det nollställas och minuten ökar med ett steg och vidare med timmar. (b) Istället för att öka med en sekund i taget så minskar man med 60 sekunder eller minuter i stöten om dessa variabler är större än 60, 60 och 24 och i samma veva justera de högre värdesiffrorna med ett steg. (c) Det absolut snabbaste och kortaste svaret blir dock med direkt heltalsdivision med och utan resträkning dvs; void oka(struct tidpunkt *t, int s){ t->s = (t->s+s)%60; t->m = (t->m+(s/60))%60; t->h = (t->h+(s/3600))%24; }
Om du har frågor om dina kommentarer, din tentamen eller något annat i kursen, tveka inte att kontakta Filip Sebek. Du kan självklart lämna åsikter anonymt i hans fack. Försök i så fall vara så detaljerad som möjligt.

Bedömningsunderlag för Tentamen CDT106

I

Uppg 1 +	G - Dugga
Uppg 1 -	
Uppg 2 +	Helt korrekt funktionsdeklaration, anrop och definition Fullt fungerande program
Uppg 2 -	
Uppg 3 +	Korrekt utskrift vilket indikerar förståelse för programflöde, komplexa nästlade loopar och förmåga att sätta sig in i kod skriven av andra. Ganska väldokumenterad exekveringsbeskrivning
Uppg 3 -	
Uppg 4 +	Nästan korrekt löst med både med och utan funktioner Arrayer är korrekt deklarerade, tilldelade och hanterade i samband med funktionsanrop. Arrayaccesser är enkel indexering
Uppg 4 -	1, programmet utgår från att största talet är minst 0. Säkrare/mer rätt att initiera till max=x[0] och börja loopen med for(i=1 . . .) 2, funktionen readvector har som inargument två vektorreferens vilket är fel enligt spec. Ska ta EN vektor i taget (anropas två gånger) 3, biggestinvector ska enligt spec funktionen returnera det största värdet i EN vektor. 4, (Vad används returvärdet till från biggestinvector och varför finns det ingen return därifrån då det är en int-funktion - en pekarreferens används ju redan)
Uppg 5 +	använder while istället för for använder sig av pekarstegring
Uppg 5 -	
Uppg 6 +	tiden lagrad som en struktur Algoritmiskt är det mesta rätt, men en hel del syntaxfel
Uppg 6 -	1, (borde delat upp programmet i något fler funktioner) 2, funktionsanropet till laggtill i main tar inte hand om returvärdet 3, i laggtilltid läses tillägget till variabeln "laggtill", som sedan plötsligt byter namn till val (?) 4, >= felskrivet 5, wraparound på 23:59:59 är inte implementerat 6, Första while-satsen i laggtilltid redundant 7, Pseudokod något för översiktlig beträffande "lägg till tid" - mera detaljer efterlyses i synnerhet i slutet på beskrivningen.
Uppg 7 +	Pseudokod inkluderat - ganska korrekt - C-kodsimplementation med flera syntaxfel
Uppg 7 -	1, fopen görs inte med den generella invariablen filename utan på den hårdkodade "filename" pga citationstecken 2, läsa fil kan inte göras med strcpy, det görs med fgets(). Strcpy() har fel ordning på argumenten 3, andra fopen måste vara i mode "a" eller "r+" eftersom "w" skriver sönder den öppnade filen 4, fprintf(skrivfil,\n) borde vara fprintf(skrivfil,"\n") - fast den behövs egentligen inte 5, fprintf(skrivfil,"%c",oppna); borde vara "%s"
Uppg 8 +	funktionen strcat används korrekt, men den implemenerade funktionen skriver ut en massa som den inte borde. Man kan därför anta att du inte har riktigt full koll på vad strcat gör med tanke på all extra kod
Uppg 8 -	1, vad händer om antalet tecken är exakt 20? 2, vad är den deklarerade struct retur? Borde det inte vara char *retur? Fast den behövs inte alls egentligen 3, While-strukturerna i respekive if-sats kan strykas helt!

BONUS

Plus-laboration	G

Bedömningsunderlag för Tentamen CDT106

I

Alla labbar klara	G

Något av följande projekt klar innan tentamen

Videobutik	---	**Altera DE2**	---
Robotik	---	**Projdok Chat**	

Sammanfattning	Listorna på missar i uppgift 4,6 och 7 ser ganska långa ut, men alla påpekandena är inte så allvarliga ur en bedömningssynpunkt. Grunderna i språket måste beskrivas som fullt tillfredställande. Dock är det en del syntaxfel som delvis skulle kunna ha hittats med en kompilator. Med bonusuppgifter från laborationer blir det därför en 4 i betyg eftersom de betygshöjande uppgifterna delvis är lösta. Och med lite mera programmeringsvana så blir man också säkrare i språket.
Betyg	**4**

Gränsvärden för betyg

Hur man får godkänt i kursen och denna tentamen står beskrivet i kursPM, men här följer ändå en lathund som lite grovt beskriver vad som krävdes på denna tentamen. Innehåller lösningsförslagen några fel kan i första hand A-uppgifter kompenseras med bonus från laborationer och färdigställda projekt. Undantagsvis kan bonus kompensera på G-uppgifter.
3 = Klarat Uppgift 1-3 samt uppgift 4 med funktioner utan några större fel.
4 = Som för betyget 3 samt ytterligare två A-moment varav minst en måste vara förvärvad från uppgift 5 eller 8
5 = Besvarat alla uppgifter utan några större fel. Om någon A-uppgift innehåller fel kan detta kompenseras med bonus.

Bedömningsfokus
Uppg1: Grundläggande I/O, villkor _loopar, Uppg2: Definition, deklaration och anrop av funktioner, Uppg3: Analys och förståelse av nästlade loopar, Uppg4: Arrayer, funktionsanrop med arrayer, funktionsbegrepp återanvändning av kod, Uppg5: Bredd på C-språket; Uppg6: Komplexitet, problemlösningsförmåga och ev. structer; Uppg7: Filhantering, Uppg8: Förmåga att läsa funktionsdokumentation, stränghantering.
Kan man visa i andra uppgifter att man behärskar något av ovanstående så har man det självklart till godo.

Övrig information

Examinatorn har försökt att hitta såväl korrekta som felaktiga delar i lösningsförslagen och även ge tips om förbättringar. Med tanke på det stora antalet prov som ska bedömas på kort tid, kan det tyvärr bli slagsida på endera typen av kommentarer vilket inte är avsikten. Det finns självklart ingen som helst avsikt att sänka någon. Programmering ÄR tyvärr svårare än de flesta anar, och det kan därför ta tid innan man behärskar tillräckligt mycket för att kunna bli godkänd. Det är examinatorns mål att alla som vill lära sig programmera ska få tillräckligt med handledning för att kunna klara kursen. Däremot kan ingen lova hur lång tid det kan ta för att nå det målet eftersom inlärningsförmåga och tidigare erfarenheter är individuella egenskaper och inlärning är heller inte en linjär process utan går för de flesta i trappsteg som är olika höga.
Den mest komplicerade uppgiften på tentan var uppgift 6 där man skulle lägga till ett antal sekunder till en vis tid. Om man lagrar tiden i en struct tidpunkt{ int h,m,s;} och tidstillägget ligger i int s; så finns det flera sätt att lösa problemet. (a) Ett enkelt men onödigt processorintensivt är att loopa tidstillägget nedåt tills det når noll. Öka sekunder för varje varv och om det passerar 60 så ska det nollställas och minuten ökar med ett steg och vidare med timmar. (b) Istället för att öka med en sekund i taget så minskar man med 60 sekunder eller minuter i stöten om dessa variabler är större än 60, 60 och 24 och i samma veva justera de högre värdesiffrorna med ett steg. (c) Det absolut snabbaste och kortaste svaret blir dock med direkt heltalsdivision med och utan resträkning dvs; void oka(struct tidpunkt *t, int s){ t->s = (t->s+s)%60; t->m = (t->m+(s/60))%60; t->h = (t->h+(s/3600))%24; }
Om du har frågor om dina kommentarer, din tentamen eller något annat i kursen, tveka inte att kontakta Filip Sebek. Du kan självklart lämna åsikter anonymt i hans fack. Försök i så fall vara så detaljerad som möjligt.

Bedömningsunderlag för Tentamen CDT106

J

Uppg 1 +	Korrekt variabeldeklaration, sekvenser av kod, selektion och loopars villkorsuttryck syntaktiskt rätt hanterat, korrekt I/O
Uppg 1 -	1, radieinläsning borde ske utanför loopen 2, if(temp<r) borde vara if(temp<=r)
Uppg 2 +	Helt korrekt funktionsdeklaration, anrop och definition Fullt fungerande program
Uppg 2 -	
Uppg 3 +	Korrekt utskrift vilket indikerar förståelse för programflöde, komplexa nästlade loopar och förmåga att sätta sig in i kod skriven av andra.
Uppg 3 -	
Uppg 4 +	1, Korrekt program 2, Korrekt beteende i programmet ur användarsynpunkt 3, Arrayer är korrekt deklarerade, tilldelade och hanterade i samband med funktionsanrop. 4, Arrayaccesser är enkel indexering i for-loopar 5, Korrekt interface mot funktionerna
Uppg 4 -	
Uppg 5 +	EJ LÖST UPPGIFT
Uppg 5 -	
Uppg 6 +	1, Lagrat tiden i en struct 2, tidsuppdatering sker genom pekare 3, korrekt påbörjad adderadtid - ej avslutad 4, pseudokod är inte tillräckligt specad - är snarare en helhetsbild över programmet - ungenfär som ett funktionsträd.
Uppg 6 -	1, Ofullständig lösning 2, I adderatid är punkt en adress. Alltså borde t.ex. punkt.tim skrivas punkt -> tim
Uppg 7 +	EJ LÖST UPPGIFT
Uppg 7 -	
Uppg 8 +	EJ LÖST UPPGIFT
Uppg 8 -	

BONUS

Plus-laboration	---
Alla labbar klara	G 080104

Något av följande projekt klar innan tentamen

Videobutik	---	**Altera DE2**	
Robotik		**Projdok Chat**	---

Sammanfattning	Alla grundläggande delar såsom I/O, selektion, iteration, variabelhantering, funktioner och arrayer är avklarade vilket är kravet för godkänt - något som du gjort med marginal.
Betyg	**3**

Bedömningsunderlag för Tentamen CDT106

J

Gränsvärden för betyg

Hur man får godkänt i kursen och denna tentamen står beskrivet i kursPM, men här följer ändå en lathund som lite grovt beskriver vad som krävdes på denna tentamen. Innehåller lösningsförslagen några fel kan i första hand A-uppgifter kompenseras med bonus från laborationer och färdigställda projekt. Undantagsvis kan bonus kompensera på G-uppgifter.
3 = Klarat Uppgift 1-3 samt uppgift 4 med funktioner utan några större fel.
4 = Som för betyget 3 samt ytterligare två A-moment varav minst en måste vara förvärvad från uppgift 5 eller 8
5 = Besvarat alla uppgifter utan några större fel. Om någon A-uppgift innehåller fel kan detta kompenseras med bonus.

Bedömningsfokus
Uppg1: Grundläggande I/O, villkor _loopar, Uppg2: Definition, deklaration och anrop av funktioner, Uppg3: Analys och förståelse av nästlade loopar, Uppg4: Arrayer, funktionsanrop med arrayer, funktionsbegrepp återanvändning av kod, Uppg5: Bredd på C-språket; Uppg6: Komplexitet, problemlösningsförmåga och ev. structer; Uppg7: Filhantering, Uppg8: Förmåga att läsa funktionsdokumentation, stränghantering.
Kan man visa i andra uppgifter att man behärskar något av ovanstående så har man det självklart till godo.

Övrig information

Examinatorn har försökt att hitta såväl korrekta som felaktiga delar i lösningsförslagen och även ge tips om förbättringar. Med tanke på det stora antalet prov som ska bedömas på kort tid, kan det tyvärr bli slagsida på endera typen av kommentarer vilket inte är avsikten. Det finns självklart ingen som helst avsikt att sänka någon. Programmering ÄR tyvärr svårare än de flesta anar, och det kan därför ta tid innan man behärskar tillräckligt mycket för att kunna bli godkänd. Det är examinatorns mål att alla som vill lära sig programmera ska få tillräckligt med handledning för att kunna klara kursen. Däremot kan ingen lova hur lång tid det kan ta för att nå det målet eftersom inlärningsförmåga och tidigare erfarenheter är individuella egenskaper och inlärning är heller inte en linjär process utan går för de flesta i trappsteg som är olika höga.
Den mest komplicerade uppgiften på tentan var uppgift 6 där man skulle lägga till ett antal sekunder till en vis tid. Om man lagrar tiden i en struct tidpunkt{ int h,m,s;} och tidstillägget ligger i int s; så finns det flera sätt att lösa problemet. (a) Ett enkelt men onödigt processorintensivt är att loopa tidstillägget nedåt tills det når noll. Öka sekunder för varje varv och om det passerar 60 så ska det nollställas och minuten ökar med ett steg och vidare med timmar. (b) Istället för att öka med en sekund i taget så minskar man med 60 sekunder eller minuter i stöten om dessa variabler är större än 60, 60 och 24 och i samma veva justera de högre värdesiffrorna med ett steg. (c) Det absolut snabbaste och kortaste svaret blir dock med direkt heltalsdivision med och utan resträkning dvs; void oka(struct tidpunkt *t, int s){ t->s = (t->s+s)%60; t->m = (t->m+(s/60))%60; t->h = (t->h+(s/3600))%24; }
Om du har frågor om dina kommentarer, din tentamen eller något annat i kursen, tveka inte att kontakta Filip Sebek. Du kan självklart lämna åsikter anonymt i hans fack. Försök i så fall vara så detaljerad som möjligt.

Bedömningsunderlag för Tentamen CDT106

K

Uppg 1 +	Till största delen korrekt program
Uppg 1 -	1, radien ska läsas in utanför loopen 2, scanf med otillåtna tecken såsom \n 3, kvadreringstecken finns inte i C. Antingen x*x eller pow(x,2) är det som gäller 4, if-satsen ska inte vara = utan >= eller möjligen ==
Uppg 2 +	korrekt funktionsanrop, returvärde och funktionsdefinition (se -)
Uppg 2 -	avslutande klammer i main på fel plats funktionen deklarerad och definierad utan returtyp (borde vara float)
Uppg 3 +	1,greppat första inre loopen som skriver ut blanksteg 2, Nästan korrekt utskrift vilket indikerar förståelse för programflöde, komplexa nästlade loopar och förmåga att sätta sig in i kod skriven av andra.
Uppg 3 -	inte greppat fulltsändigt att den andra inre loopen ska minska i antal varv pga ökat initialvärde och minskat max-värde. Bommat nyradtecken
Uppg 4 +	1, Skrivit hela programmet i main() utan funktioner
Uppg 4 -	1, tilldelning i scanf sker i max-poisitionen hela tiden borde vara scanf("%d",&arr1[i]); Dito vid 2, programmet utgår från att största talet är minst 0. Säkrare/mer rätt att initiera till storsta1=arr1[0] och börja loopen med for(i=1 . . .) 3, MAX ska heta LENGTH enligt uppgiften 4, Arrayhantering med många brister
Uppg 5 +	while funktion som jämför de två vektorerna
Uppg 5 -	1, anropet görs med stor(int array1[max], . . .); vilket är fel: ta bort typen och även klamrarna med index. Nu skickas ett värde utanför arrayen till funktionen - inte hela vektorn 2, felaktig tildelning som i uppg 4
Uppg 6 +	Lagrar tiden i struct - man använder sig aldrig av strukturen - bara temporära variabler Ofullständigt program och ofullständig pseuokod
Uppg 6 -	1, orimilig test if(minuter == 0 && minuter<60) dvs minuter måste vara 0. Dessutom sker testen innan värdet är inmatat. Förmodligen borde det varit en do{}while 2, ingen many 3, ofullständig kod i main() - saknas t.ex. funktionsanrop 4I funktionen står det som inparameter (struct tidt *). Borde vara (struct tidt *tid) 5, adderar sekunder med ny, men tilldelar(lagrar) algrig uträkningen någonstans
Uppg 7 +	EJ LÖST UPPGIFT
Uppg 7 -	
Uppg 8 +	EJ LÖST UPPGIFT
Uppg 8 -	

BONUS

Plus-laboration	---
Alla labbar klara	---

Något av följande projekt klar innan tentamen

Videobutik	---	**Altera DE2**	

Bedömningsunderlag för Tentamen CDT106

K

Robotik		**Projdok Chat**	---

Sammanfattning	Alla uppgifterna innehåller många syntaktiska fel vilket indikerar på en viss ovana i programmering (vilket även bekräftas med att alla laborationer inte är gjorda än). Formellt sett så blir du underkänd för att du inte skrivit något korrekt program och för att det viktiga avsnittet om arrayer inte är avklarat. Mycket av den logiska tankenstrukturen finns så med lite mera övning så kommer du säkert att nå målen för godkänt.
Betyg	**U**

Gränsvärden för betyg

Hur man får godkänt i kursen och denna tentamen står beskrivet i kursPM, men här följer ändå en lathund som lite grovt beskriver vad som krävdes på denna tentamen. Innehåller lösningsförslagen några fel kan i första hand A-uppgifter kompenseras med bonus från laborationer och färdigställda projekt. Undantagsvis kan bonus kompensera på G-uppgifter.
3 = Klarat Uppgift 1-3 samt uppgift 4 med funktioner utan några större fel.
4 = Som för betyget 3 samt ytterligare två A-moment varav minst en måste vara förvärvad från uppgift 5 eller 8
5 = Besvarat alla uppgifter utan några större fel. Om någon A-uppgift innehåller fel kan detta kompenseras med bonus.

Bedömningsfokus
Uppg1: Grundläggande I/O, villkor _loopar, Uppg2: Definition, deklaration och anrop av funktioner, Uppg3: Analys och förståelse av nästlade loopar, Uppg4: Arrayer, funktionsanrop med arrayer, funktionsbegrepp återanvändning av kod, Uppg5: Bredd på C-språket; Uppg6: Komplexitet, problemlösningsförmåga och ev. structer; Uppg7: Filhantering, Uppg8: Förmåga att läsa funktionsdokumentation, stränghantering.
Kan man visa i andra uppgifter att man behärskar något av ovanstående så har man det självklart till godo.

Övrig information

Examinatorn har försökt att hitta såväl korrekta som felaktiga delar i lösningsförslagen och även ge tips om förbättringar. Med tanke på det stora antalet prov som ska bedömas på kort tid, kan det tyvärr bli slagsida på endera typen av kommentarer vilket inte är avsikten. Det finns självklart ingen som helst avsikt att sänka någon. Programmering ÄR tyvärr svårare än de flesta anar, och det kan därför ta tid innan man behärskar tillräckligt mycket för att kunna bli godkänd. Det är examinatorns mål att alla som vill lära sig programmera ska få tillräckligt med handledning för att kunna klara kursen. Däremot kan ingen lova hur lång tid det kan ta för att nå det målet eftersom inlärningsförmåga och tidigare erfarenheter är individuella egenskaper och inlärning är heller inte en linjär process utan går för de flesta i trappsteg som är olika höga.
Den mest komplicerade uppgiften på tentan var uppgift 6 där man skulle lägga till ett antal sekunder till en vis tid. Om man lagrar tiden i en struct tidpunkt{ int h,m,s;} och tidstillägget ligger i int s; så finns det flera sätt att lösa problemet. (a) Ett enkelt men onödigt processorintensivt är att loopa tidstillägget nedåt tills det når noll. Öka sekunder för varje varv och om det passerar 60 så ska det nollställas och minuten ökar med ett steg och vidare med timmar. (b) Istället för att öka med en sekund i taget så minskar man med 60 sekunder eller minuter i stöten om dessa variabler är större än 60, 60 och 24 och i samma veva justera de högre värdesiffrorna med ett steg. (c) Det absolut snabbaste och kortaste svaret blir dock med direkt heltalsdivision med och utan resträkning dvs; void oka(struct tidpunkt *t, int s){ t->s = (t->s+s)%60; t->m = (t->m+(s/60))%60; t->h = (t->h+(s/3600))%24; }
Om du har frågor om dina kommentarer, din tentamen eller något annat i kursen, tveka inte att kontakta Filip Sebek. Du kan självklart lämna åsikter anonymt i hans fack. Försök i så fall vara så detaljerad som möjligt.

Bedömningsunderlag för Tentamen CDT106

L

Uppg 1 +	Korrekt variabeldeklaration, sekvenser av kod, selektion och loopars villkorsuttryck syntaktiskt rätt hanterat
Uppg 1 -	variabel z är oinitierad i första while-testet kooridinat-testet på punkt (-1.0 , +1.0) kommer få programmet att terminera
Uppg 2 +	Korrekt funktionsanrop med inparametrar och tilldelning av returvärde. Funktionen korrekt definierad och deklarerad
Uppg 2 -	
Uppg 3 +	Inte rätt
Uppg 3 -	Går ej att bedöma hur mycket som är greppat - för lite underlag. Det borde vara 7 nyradstecken med tanke på yttre loopen - i lösningsförslaget finns bara 6 rader
Uppg 4 +	Till största delen korrekt rent konceptuellt om det varit en G-lösning Följer de specade funktionsinterfacen korrekt arrayhantering; deklaration & tilldelning arrayhanteringen sker med forsatser och enkel indexering. BiggestInVector korrekt implementerad
Uppg 4 -	1, i readVector skickas a[] in, men med * eller [] för mycket. Antingen ska det vara int *a eller int a[] - inte både och för det blir en dubbelpekare. Vidare så glöms denna inparameter helt bort och lagring sker i den lokala arrayen arr !?!?
Uppg 5 +	EJ LÖST UPPGIFT
Uppg 5 -	
Uppg 6 +	tidpunkt korrekt lagrad i en struct väl modulariserad kod korrekt hanterad parameterpassning med returvärden eller pekare till struct nästan helt korrekt funktion i Nytid mha en iterativ metod
Uppg 6 -	- om en tidpunkt kommer in som en pekare så måste man använda -> istället för punkt för att accessa fälten i posten. - i funktionen NyTid uppdateras minuter och timmar men inte tiden.sek (!)
Uppg 7 +	Lite inledande onödigheter, men i övrigt korrekt program
Uppg 7 -	1, du behöver inte anta att filnamnet innehåller ett nyradtecken - det spelar faktiskt ingen roll utan funkar ändå… 2, "filename" är ett hårdkodat filnamn. Ta bort citationstecknena och du åfr användning av variabelnamnet filename
Uppg 8 +	EJ LÖST UPPGIFT
Uppg 8 -	

BONUS

Plus-laboration	---
Alla labbar klara	---

Något av följande projekt klar innan tentamen

Videobutik	---	**Altera DE2**	
Robotik		**Projdok Chat**	---

Sammanfattning	Kravet är att G-uppgifterna är korrekta, och uppgift 3 är tämligen olöst. Och även om det hade varit okej så är det synd att du inte löste uppg 5 eller 8 så hade du kunnat få högre betyg. Jag erbjuder dig komplettering för att få betyget 3 (men ingen plussning)

Bedömningsunderlag för Tentamen CDT106

L

Betyg	**3**

Gränsvärden för betyg

Hur man får godkänt i kursen och denna tentamen står beskrivet i kursPM, men här följer ändå en lathund som lite grovt beskriver vad som krävdes på denna tentamen. Innehåller lösningsförslagen några fel kan i första hand A-uppgifter kompenseras med bonus från laborationer och färdigställda projekt. Undantagsvis kan bonus kompensera på G-uppgifter.
3 = Klarat Uppgift 1-3 samt uppgift 4 med funktioner utan några större fel.
4 = Som för betyget 3 samt ytterligare två A-moment varav minst en måste vara förvärvad från uppgift 5 eller 8
5 = Besvarat alla uppgifter utan några större fel. Om någon A-uppgift innehåller fel kan detta kompenseras med bonus.

Bedömningsfokus
Uppg1: Grundläggande I/O, villkor loopar, Uppg2: Definition, deklaration och anrop av funktioner, Uppg3: Analys och förståelse av nästlade loopar, Uppg4: Arrayer, funktionsanrop med arrayer, funktionsbegrepp återanvändning av kod, Uppg5: Bredd på C-språket; Uppg6: Komplexitet, problemlösningsförmåga och ev. structer; Uppg7: Filhantering, Uppg8: Förmåga att läsa funktionsdokumentation, stränghantering.
Kan man visa i andra uppgifter att man behärskar något av ovanstående så har man det självklart till godo.

Övrig information

Examinatorn har försökt att hitta såväl korrekta som felaktiga delar i lösningsförslagen och även ge tips om förbättringar. Med tanke på det stora antalet prov som ska bedömas på kort tid, kan det tyvärr bli slagsida på endera typen av kommentarer vilket inte är avsikten. Det finns självklart ingen som helst avsikt att sänka någon. Programmering ÄR tyvärr svårare än de flesta anar, och det kan därför ta tid innan man behärskar tillräckligt mycket för att kunna bli godkänd. Det är examinatorns mål att alla som vill lära sig programmera ska få tillräckligt med handledning för att kunna klara kursen. Däremot kan ingen lova hur lång tid det kan ta för att nå det målet eftersom inlärningsförmåga och tidigare erfarenheter är individuella egenskaper och inlärning är heller inte en linjär process utan går för de flesta i trappsteg som är olika höga.
Den mest komplicerade uppgiften på tentan var uppgift 6 där man skulle lägga till ett antal sekunder till en vis tid. Om man lagrar tiden i en struct tidpunkt{ int h,m,s;} och tidstillägget ligger i int s; så finns det flera sätt att lösa problemet. (a) Ett enkelt men onödigt processorintensivt är att loopa tidstillägget nedåt tills det når noll. Öka sekunder för varje varv och om det passerar 60 så ska det nollställas och minuten ökar med ett steg och vidare med timmar. (b) Istället för att öka med en sekund i taget så minskar man med 60 sekunder eller minuter i stöten om dessa variabler är större än 60, 60 och 24 och i samma veva justera de högre värdesiffrorna med ett steg. (c) Det absolut snabbaste och kortaste svaret blir dock med direkt heltalsdivision med och utan resträkning dvs; void oka(struct tidpunkt *t, int s){ t->s = (t->s+s)%60; t->m = (t->m+(s/60))%60; t->h = (t->h+(s/3600))%24; }
Om du har frågor om dina kommentarer, din tentamen eller något annat i kursen, tveka inte att kontakta Filip Sebek. Du kan självklart lämna åsikter anonymt i hans fack. Försök i så fall vara så detaljerad som möjligt.

Bedömningsunderlag för Tentamen CDT106

M

Uppg 1 +	G - Dugga
Uppg 1 -	
Uppg 2 +	G - Dugga
Uppg 2 -	
Uppg 3 +	Korrekt utskrift vilket indikerar förståelse för programflöde, komplexa nästlade loopar och förmåga att sätta sig in i kod skriven av andra.
Uppg 3 -	
Uppg 4 +	1,Program med en del syntaktiska fel 2, Arrayer är korrekt tilldelade 3, Arrayaccesser är enkel indexering i for-loopar
Uppg 4 -	1, funktionsanropen i main() innehåller felaktiga klamrar 2, arrayerna är mycket märkligt deklarerarde av storleken 0 och sedan omdeklarerade 3, funktionen biggestvector kommer inte att hitta största talet
Uppg 5 +	
Uppg 5 -	En variant på en skickad pekare med funktionsanrop men funktionerna är inte definierade
Uppg 6 +	Ofullständig kod - endast menyn redovisas
Uppg 6 -	1, Beskriver inte hur tiden ska lagras - som sträng, heltal, struct? 2, Funktionsanrop utan parametrar 3, Ofullständig pseudokod
Uppg 7 +	EJ LÖST UPPGIFT
Uppg 7 -	
Uppg 8 +	EJ LÖST UPPGIFT
Uppg 8 -	

BONUS

Plus-laboration	G
Alla labbar klara	G 071206

Något av följande projekt klar innan tentamen

Videobutik	---	**Altera DE2**	---
Robotik	---	**Projdok Chat**	

Sammanfattning	Grundläggande I/O, loopar, selektion och funktionshantering finns, men alla delar i arrayhantering funkar inte Det är en pyttelitenliten bit kvar till godkänt så med en komplettering så blir det godkänt.
Betyg	**3**

Bedömningsunderlag för Tentamen CDT106

M

Gränsvärden för betyg

Hur man får godkänt i kursen och denna tentamen står beskrivet i kursPM, men här följer ändå en lathund som lite grovt beskriver vad som krävdes på denna tentamen. Innehåller lösningsförslagen några fel kan i första hand A-uppgifter kompenseras med bonus från laborationer och färdigställda projekt. Undantagsvis kan bonus kompensera på G-uppgifter.
3 = Klarat Uppgift 1-3 samt uppgift 4 med funktioner utan några större fel.
4 = Som för betyget 3 samt ytterligare två A-moment varav minst en måste vara förvärvad från uppgift 5 eller 8
5 = Besvarat alla uppgifter utan några större fel. Om någon A-uppgift innehåller fel kan detta kompenseras med bonus.

Bedömningsfokus
Uppg1: Grundläggande I/O, villkor _loopar, Uppg2: Definition, deklaration och anrop av funktioner, Uppg3: Analys och förståelse av nästlade loopar, Uppg4: Arrayer, funktionsanrop med arrayer, funktionsbegrepp återanvändning av kod, Uppg5: Bredd på C-språket; Uppg6: Komplexitet, problemlösningsförmåga och ev. structer; Uppg7: Filhantering, Uppg8: Förmåga att läsa funktionsdokumentation, stränghantering.
Kan man visa i andra uppgifter att man behärskar något av ovanstående så har man det självklart till godo.

Övrig information

Examinatorn har försökt att hitta såväl korrekta som felaktiga delar i lösningsförslagen och även ge tips om förbättringar. Med tanke på det stora antalet prov som ska bedömas på kort tid, kan det tyvärr bli slagsida på endera typen av kommentarer vilket inte är avsikten. Det finns självklart ingen som helst avsikt att sänka någon. Programmering ÄR tyvärr svårare än de flesta anar, och det kan därför ta tid innan man behärskar tillräckligt mycket för att kunna bli godkänd. Det är examinatorns mål att alla som vill lära sig programmera ska få tillräckligt med handledning för att kunna klara kursen. Däremot kan ingen lova hur lång tid det kan ta för att nå det målet eftersom inlärningsförmåga och tidigare erfarenheter är individuella egenskaper och inlärning är heller inte en linjär process utan går för de flesta i trappsteg som är olika höga.
Den mest komplicerade uppgiften på tentan var uppgift 6 där man skulle lägga till ett antal sekunder till en vis tid. Om man lagrar tiden i en struct tidpunkt{ int h,m,s;} och tidstillägget ligger i int s; så finns det flera sätt att lösa problemet. (a) Ett enkelt men onödigt processorintensivt är att loopa tidstillägget nedåt tills det når noll. Öka sekunder för varje varv och om det passerar 60 så ska det nollställas och minuten ökar med ett steg och vidare med timmar. (b) Istället för att öka med en sekund i taget så minskar man med 60 sekunder eller minuter i stöten om dessa variabler är större än 60, 60 och 24 och i samma veva justera de högre värdesiffrorna med ett steg. (c) Det absolut snabbaste och kortaste svaret blir dock med direkt heltalsdivision med och utan resträkning dvs; void oka(struct tidpunkt *t, int s){ t->s = (t->s+s)%60; t->m = (t->m+(s/60))%60; t->h = (t->h+(s/3600))%24; }
Om du har frågor om dina kommentarer, din tentamen eller något annat i kursen, tveka inte att kontakta Filip Sebek. Du kan självklart lämna åsikter anonymt i hans fack. Försök i så fall vara så detaljerad som möjligt.