TDP004 - Datortenta (DAT2)

2012-12-19

Regler

- All kod som skickas in för rätting skall kompilera och vara väl testad.
- Inga elektroniska hjälpmedel får medtas. Mobiltelefon ska vara avstängd och ligga i jacka eller väska.
- Inga ytterkläder eller väskor vid skrivplatsen.
- Student får lämna salen tidigast en timme efter tentans start.
- Vid toalettbesök eller rökpaus ska pauslista utanför salen fyllas i.
- All form av kontakt mellan studenter under tentans gång är strängt förbjuden.
- Böcker och anteckningssidor kan komma att granskas av tentavakt i samband med tentans start samt under tentans gång.
- Frågor om specifika uppgifter eller om tentan i stort ska ställas via tentasystemets kommunikationsklient.
- Systemfrågor kan ställas till assistent i sal genom att räcka upp handen.
- Ingen uppgift rättas efter tentatidens slut.
- Ingen uppgift kan kompletteras under tentans sista kvart.
- En praktisk uppgift kan kompletteras för högre poäng tills den är poängsatt med "Klar". "Klar" sätts vid bedömningen att ingen nämnvärd förbättring skett sedan tidigare inskickning.
- En teoretisk uppgift kan inte kompletteras. Undantagsvis kan examinator be om förtydligande av ett svar.
- Kompilerande kod är ett grundkrav för poängsättning. Kod som inte kompilerar skall kommenteras ut och kommenteras före inskickning.

Antal uppgifter	9
Totalt antal poäng	20
Hjälpmedel	En C++-bok (t.ex. C++ Primer 5th ed.) En A4-sida med egna anteckningar

Betygssättning

Tentan är uppdelad i två delar, en teoretisk och en praktisk. Delarna är värda 5 respektive 15 poäng och för godkänt betyg krävs minst 11 poäng totalt. Poäng som krävs för de olika betygen kan ses i tabell 1.

Poängsumma	Betyg
0 – 10	U
11 – 14	3
15 – 17	4
18 – 20	5

Tabell 1: Poängfördelning för betygssättning

Bonus från labserien

Avklarad dugga i kursen ger bonus i form av en tillgodoräknad uppgift i praktiska delen. Avklarade deadlines i kursen ger bonus i form av extra poäng. All bonus ges endast under den första ordinarie tentan i samband med kursen. För denna tentamen får student uppgifter tillgodoräknade enligt tabell 2. Vi kontrollerar bonus under tentans gång och tilldelar rätt poäng. Tentasystemet uppmärksammar student när ändringen sker.

Antal avklarade deadlines	Extra poäng
2 – 3	1p
4 – 5	2p
6 – 7	3p

Tabell 2: Bonus för deadlines

Information

Inloggning

Logga in på tentakontot med följande användaruppgifter:

```
Användarnamn: examx
Lösenord:     kluring1
```

Följ menyvalen så långt det går tills du ska mata in ett engångslösenord. Tag fram ditt LiU-kort och visa det för tentavakten för att få detta lösenord.

När du är inloggad är det viktigt att du startar tentaklienten genom att högerklicka på bakgrunden och välja “tentaklient” i menyn.

Avslutning

Tryck på knappen märkt “exit” i menyn längst nere på skärmen och välj ok. Vänta ett tag och tryck sedan på knappen “Avsluta tentamen” när det är möjligt. När detta är gjort är det omöjligt att logga in igen. Lämna inte datorn förrän du ser den vanliga inloggningsskärmen.

Teoretisk del

Dessa uppgifter kan *INTE* kompletteras. Skriv ditt svar på en fil med namnet *teoriN.txt* (där *N* är uppgiftsnumret) och skicka in filen för rättning. En fil och en inskickning för varje uppgift.

1. En variabel vet du sedan gammalt att den har ett namn, ett värde och en datatyp (tolkning). Med hänsyn till detta, vad är en pekare? [1 p]

   Svar:

   En pekare är en vanlig variabel som lagrar en adress. Datatypen är alltså en adress, och värdet tolkas då som adressen till någon annan variabel.

2. Klassen Bil i nedan exempel står i förhållande till klasserna Hjul, Kaross, Nyckel och Fordon enligt nedan kod. Förklara kortfattat med hjälp av exemplet innebörden av förhållandena Arv, Aggregation, Komposition och Association. 1-2 rätt ger en poäng. 3-4 rätt ger 2 poäng. [2 p]

```
class Bil : public Fordon
{
public:
  starta(Nyckel nyckel);
private:
  Hjul hjul[4];
  Kaross kaross;
};
```

   Svar:

   Arv: En bil är ett fordon; class Bil : public Fordon;
   Komposition: En Bil består (viktig del) av en kaross; Kaross kaross;
   Aggregation: En Bil har ett antal hjul; Hjul hjul[4];
   Association: En Bil känner till en Nyckel; starta(Nyckel nyckel);

3. En listklass har två medlemsfunktioner “contains” för att kontrollera om ett värde finns i listan eller om alla värden i en annan lista finns i listan. Vilka två deklarationer är bäst? [1 p]

   (a)
```
bool contains(int value);
bool contains(List list);
```

   (b)
```
bool contains(int value) const;
bool contains(List list) const;
```

   (c)
```
bool contains(int value);
bool contains(List const& list) const;
```

   (d)
```
bool contains(int value) const;
bool contains(List const& list) const;
```

   (e)
```
bool contains(int const& value) const;
bool contains(List const& list) const;
```

(f)
```
bool contains(int const value);
bool contains(List list);
```

Svar:

Inparameter av härledd typ skall tas som const-referens, inparameter av grundtyp kopieras in och medlemsfunktioner som inte ändrar medlemsvariabler bör vara const.

```
bool contains(int value) const;
bool contains(List const& list) const;
```

4. Vad är det för skillnad mellan en klass och en instans? [1 p]

Svar:

En instans är en variabel av klasstypen (med namn, värde). Klassen är bara en beskrivning av hur variabeln är uppbyggd (datatypen).

Praktisk del

En uppgifter kan kompletteras tills den är poängsatt som "Klar". Poängen inom hakparenteser till höger anger vad maxpoängen för hela uppgiften är värd. Poängen inom vanliga parenteser till vänster anger vad motsvarande speciella krav värt. Du skall alltid lösa så mycket av uppgiften du bedömer att du klarar innan du skickar in.

Kompilering mm

Använd aliaset *g++11* för att kompilera med C++11-standarden.
Använd aliaset *w++11* för att även få bra varningar.
Använd aliaset *cpp11* för att även filtrera med gccfilter.
Använd *bcheck ./a.out* för att leta minnesläckor.

5. En lärare i matematik har betygsatt elevernas kunskaper inom följande fem områden: problemlösning, begrepp, metoder, resonemang och tillämpning. Betyget på varje område är satt enligt de kriterier som fastställts av Skolverket för varje betygsteg. Kriterierna finns för högsta betyg A, mellannivån C och godkändnivån E. Nås inte kriterierna sätts underkänt betyg F. Betyget på varje område för varje elev är sparat på filen *given_files/grades.txt*, en elev per rad. [4 p]

   Nu skall läraren sammanställa delbetygen till ett slutbetyg. Detta görs enligt Skolverket enligt följande regler. Det lägsta av de fem delbetygen räknas som grundbetyg. Om minst tre delbetyg är högre än grundbetyget sätts slutbetyget ett halvsteg högre (E höjs till D och C höjs till B). Annars gäller grundbetyget som slutbetyg. Speciellt gäller att betyg F aldrig kan höjas. Exempel:

```
Delbetygen A A A A F ger slutbetyget F
Delbetygen A C C A E ger slutbetyget D
Delbetygen C E C E C ger slutbetyget D
Delbetygen A A C C A ger slutbetyget B
Delbetygen A C A C C ger slutbetyget C
Delbetygen A A A A A ger slutbetyget A
```

   Du kan anta att endast betygen A,C,E,F förekommer i filen (eller inmatningen).

   Mål: Skriv ett program som läser in en fil med delbetyg och skriver ut slutbetyget för varje elev. Listan skall vara sorterad efter betyg. Filen *given_files/grades.txt* visar hur en fil med delbetyg ser ut, och filen *given_files/final_grades.txt* visar hur läraren vill att resultatet skall bli för denna fil.

   (1p) Denna del räknas som uppfylld om "Mål" är uppfyllt istället. Skriv ett program som skriver ut filinnehållet på skärmen. Filnamnet skall kunna anges vid programkörning.

   (1p) Denna del räknas som uppfylld om "Mål" är uppfyllt istället. Skriv ett program som läser in fem delbetyg från std::cin till lämplig datastruktur och skriver ut slutbetyget.

   (1p) Du använder en STL-algoritm med lambdafunktion som del av din lösning.

Svar:

Se *set_grades.cc*.

6. *Följande uppgift tillgodoräknas om du klarat duggan.* [2 p]

   En professor i statistik har en stor myntsamling. Varje mynt har ett årtal, ett nominellt värde (värdet skrivet på myntet) och ett försäljningsvärde (om det säljs idag). Dessutom kan ett mynt singlas (singla slant, coin toss) och ger då som resultat "krona" eller "klave" ("head" or "tail").

   Professorn har upptäckt att 52% av befolkningen felaktigt tror att sidan med kungens huvud är "klave" och sidan med en krona "krona". Detta är fel eftersom "krona" syftar på begreppet "rikets styre", dvs kungen. Med detta som grund utarbetar nu professorn en strategi för slantsingling.

   Professorn satsar alltid myntets nominella värde på att "krona" kommer upp. Om kungens huvud kommer upp övertygar han alltid motspelaren att denna förlorat genom att hänvisa till bevis (wikipedia mm). Om andra sidan kommer upp så låter han motspelaren förlora om denna tror sig förlorat.

   Mål: Skriv ett program som räknar ut hur mycket professorn vinner på 1000000 slantsinglingar (med olika personer). Myntet som används skapas av huvudprogrammet. Huvudprogrammet frågar användaren om de data som behövs för att skapa ett mynt.

   (1p) Implementera en klass som representerar ett mynt med de egenskaper som beskrevs ovan. Programmeraren som använder klassen skall kunna skapa många olika mynt med olika egenskaper, men inte kunna ändra egenskaperna efter ett mynt är skapat. Du behöver bara ta med de delar som huvudprogrammet behöver använda.

   (1p) Du simulerar professorns strategi korrekt.

   Svar:

   Se *coin_toss.cc*.

7. Skriv ett program som läser in ett månadsnummer (1-12) och skriver ut motsvarande månadsnamn. Programmet skall ha fullständig felkontroll och fråga efter månadsnummret tills användaren får till rätt inmatning. Månadernas namn skall lagras och hämtas från en STL-container. Det är inte tillåtet att använda if-satser eller switch-satsen för att skriva ut rätt månadsnamn. [2 p]

   Svar:

   Se *get_month.cc*.

8. Andersson har startat företaget Vinstsnurren AB och har som Chef anställt Pettersson som karuselloperatör tillsvidare. Dessutom hyrs Lundström in ett antal timmar varje månad för diverse reparationer. [3 p]

   En tillsvidareanställd får 24000 i fast månadslön minus 1000kr för varje sjukdag. En timanställd får 180kr per timme. Chefen tar ut 30000 i månaden plus en bonus på 10% av omsättningen under månaden.

   Som de flesta vet är det "Andersson, Pettersson, Lundström och Jag". Den som är "Jag" har börjat skapa ett enkelt löneberäkningsprogram till Vinstsnurren AB. Så långt har programmet en basklass för anställda, se *given_files/vinstsnurren_given.cc*.

   Mål: Huvudprogrammet skall varje månad gå igenom en lista med anställda och skriva ut lön och namn för var och en. Respektive klass har hand om de data som är samma från månad till månad, och frågar om de uppgifter som saknas för att beräkningen skall kunna genomföras.

   (1p) Du har korrekta subklasser för tillsvidareanställd, timanställd och chef.

   (1p) Löneberäkningen sker korrekt. Löneadministratören matar alltid in korrekta data så du behöver inte felkontrollera inmatningar.

   (1p) Du sköter minneshanteringen helt korrekt.

   Svar:

   Se *vinstsnurren_ab.cc*.

9. Programkoden för implementationen av en stack och ett testprogram är given på filen *given_files/stack_given.cc*. [4 p]

   Mål: Implementera kopieringskonstruktor och destruktor för klassen. Dessa skall se till att djupa kopior görs och att inget minne läcker. Det ingår att se till att alla funktioner testas ordentligt.

   (2p) Både kopieringskonstruktor och destruktor är korrekta. Poäng ges endast för båda.

   (1p) Dela upp koden i headerfil och implementationsfil för stacken, samt testprogram.

   (1p) Implementera tilldelningsoperator för klassen utan att duplicera någon kod från kopieringskonstruktor eller destruktor. Det är tillåtet att lägga till privata medlemsfunktioner om så önskas.

   Svar:

   Se *stack_good.cc*.