Tentamen i TDP004
Objektorienterad Programmering
Praktisk del

Datum:	2010-04-07
Tid:	8-12
Plats:	SU-salar i B-huset.
Jour:	Per-Magnus Olsson, tel 285607 Jourhavande kommer att besöka skrivsalarna ungefär varje timme under skrivtiden.
Hjälpmedel:	Teoretisk del: Inga. Praktisk del: Den C++ information som finns i systemet.
Betygsättning:	Max antal poäng: 44 med 22 poäng vardera på teori och praktikdel. Poäng **Betyg** 38-44 **5** 31-37 **4** 24-30 **3** 0-23 **U**
Anvisningar:	Börja med den teoretiska delen. När du är klar med den lämnar du in den och får den praktiska delen. När du har lämnat in den teoretiska delen kan du inte återvända till den. Skriv svaret på varje uppgift på ett separat blad. Uppgifterna är inte ordnade i svårighetsgrad. En kort hjälptext till Eclipse finns på sid 2-3.

Lycka till!

Kort Eclipsehjälp

Om du inte är bekant med Eclipse rekommenderas att du läser igenom följande guide.

Att komma igång
Skriv eclipse & i terminal. Efter viss väntetid kommer ett fönster att visas. Välj ”C++ Perspective” i övre högra hörnet av fönstret. ”C++ Perspective” finns under ”Other”.

Välj ”Managed Make C++ Project” som din typ av projekt. Gör ett projekt för varje uppgift, om inte annat sägs i uppgiften.

Du måste även skapa en fil som har funktionen `main` definierad. Det går bra att skapa en .cpp-fil och där definiera funktionen `main` enligt nedan, vilken blir det som operativsystemet anropar du när exekverar programmet.

```
int main()
{
  //Övriga anrop görs här.
  return 0;
}
```

Om du inte skapar en main-funktion så kommer kompilatorn att säga att funktionen main saknas och att ett allvarligt symbolreferensfel har inträffat.

Ovanstående är viktigt för att Eclipse ska bete sig korrekt med avseende på kompilering etc.

Inkludera eventuella givna filer
För att använda programfiler som finns i katalogen given_files, kopiera dem till ditt workspace. Högerklicka sedan på ”C/C++ Project” vilken normalt är ett vertikalt fönster finns längst till vänster i huvudfönstret. Välj sedan ”Refresh” i den meny som visas. De inkluderade visas nu tillsammans med dina egna filer. Programfilerna kan nu inkluderas som vanligt.
Datafiler behöver endast kopieras till ditt workspace.

Kompilera
I ”C/C++ Project”, högerklicka på projektet och välj ”Build” i menyn. Som default är Eclipse inställt på att automatiskt kompilera under tiden du programmerar.

Köra program
I ”C/C++ Project”, välj ”Runa as”->”Run as local C/C++ application”.
Det går även att välja ”Run”-knappen i menyn längst upp i fönstret.

Automatisk kompilering
Som default är Eclipse inställt på att kompilera alla filer så snart du har slutat skriva. Om du vill avaktivera den funktionen, gå till ”Project”-menyn och klicka på texten ”Build Automatically”.

Ta bort alla kompilerade filer
Om du vill ta bort alla kompilerade filer (källkodsfiler är opåverkade): gå till ”Project”-menyn och välj ”Clean”.

Om fönster inte visas
Om konsollen med resultatet av programkörningar inte visas, så kan du aktivera visningen av fönstret genom att gå att ”Project”-menyn, och välja ”Show vindow” och sedan välja ”Console”. Om ”Problems”-fönstret som visar eventuella komplieringsfel inte visas, gör samma sak men välj ”Problems” istället.

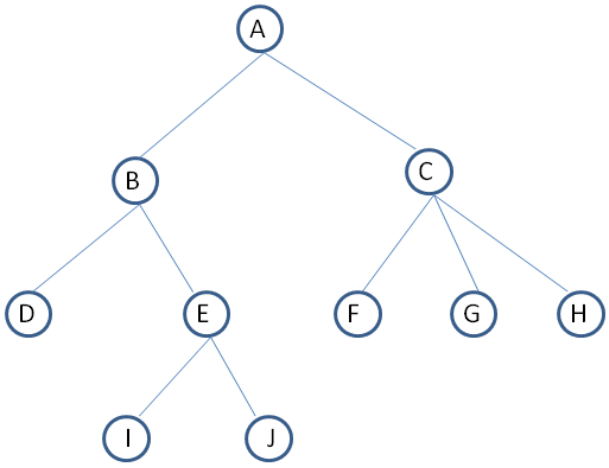
Om Eclipse verkar ha låst sig
Prova att växla mellan olika fönster med Alt-Tab. Ibland hamnar pop-up-fönster som väntar på din input bakom huvudfönstret. Om det inte fungerar kanske du måste stänga Eclipse från konsollen och starta om.

TDP004 Objektorienterad Programmering
Praktisk del

1. a) Skriv en basklass `Shape`, samt tre subklasser till denna. De tre subklasserna ska ha namnen `Pyramid`, `Rectangular_box` och `Ellipsoid`. Basklassen har tre olika medlemsvariabler av typen `double`: `width`, `depth` och `height`. Värden på dessa tas som inparametrar till subklassernas konstruktorer. Implementera även funktionen `virtual double calculate_volume() const = 0`, vilken beräknar och returnerar formens volym med hjälp av nedanstående formler och medlemsvariablerna. För `pyramid` är volymen V = (width* depth*height) / 3, för `rectangular_box` är volymen V = (width* depth*height) och för `ellipsoid` är $V = \frac{4\pi\ (width*depth*height)}{3}$. Konstanten π kan i det här fallet uppskattas till 3,14. (9p)

   b) Gör ett testfall där du visar att polymorfismen fungerar. (3p)

2. Träd är viktiga datastrukturer inom programmering och används för en mängd olika ändamål. Ett träd består av noder och bågar och ett träd måste ha precis en rotnod, vilken finns ”överst i trädet”. De flesta noder har flera barn, vilket är benämningen på de noder som finns ”under” föräldranoden i trädet. Noder som inte har några barn kallas löv. I bilden nedan är nod `A` rotnod, och noderna `D, F, G, H, I` och `J` är löv. Övriga noder är interna noder, d.v.s. varken rotnod eller löv.



   I `given_files` finns klasserna `Node` och `Tree`. `Node` implementerar noderna och `Tree` har en medlemsfunktion `Node* build_tree()` som skapar ett träd enligt bilden och returnerar en pekare till rotnoden.

   Den uppgift är att skriva en funktion som hittar en väg från rotnoden till en given målnod, och returnerar denna i en `std::vector<std::string>`, där innehållet i varje `string` är nodens namn. Vägen ska lagras i ordning, med rotnoden först och målnoden sist. Exempel: om funktionen anropas med målnod `J` så ska svaret vara en `vector` med fyra strängar, varav den första innehåller `A`, den andra strängen

innehåller `B,` den tredje `E` och den fjärde `J` (hela vägen är `A, B, E, J`). Om den givna noden inte finns i trädet ska en tom `vector` returneras.

Funktionen ska vara generell så att den klarar av andra träd än det ovanstående. Du väljer själv hur och var du implementerar din funktion: i `Node`, `Tree` eller i någon annan klass som du skapar själv. Plocka själv ut någon nod från klassen `Tree` som målnod för testning av din algoritm. Tips: med rekursion blir uppgiften lättare. (4p)

3. Katalogen given_files finns två filer med namnen `person_info.cpp/h`. Dessa är tänkta att spara information om en person, dennes adress m.m. Tyvärr har programmeraren gjort vissa fel, både modelleringsfel och missar i implementationen. Din uppgift är att hitta och rätta felen. När du ändrar något, skriv en kommentar vad du har ändrat och varför. Om du hittar saker som du skulle vilja göra om helt men inte hinner med, förklara vad det är, varför din ändring behövs och hur du tycker att det bör fungera. Uppgiften erbjuder många olika möjligheter, poäng utdelas för vettig förändring med vettig motivering. (6p)