Försättsblad till skriftlig tentamen vid Linköpings Universitet

Datum för tentamen	*2010-12-20*
Sal	*SU17-18*
Tid	*14-18*
Kurskod	*TDP004*
Provkod	*DAT2*
Kursnamn/benämning	*Objekt-orienterad programmering*
Institution	*IDA*
Antal uppgifter som ingår i tentamen	*Teoretisk del: 5* *Praktisk del: 2*
Antal sidor på tentamen (inkl. försättsbladet)	*Teoretisk del: 4* *Praktisk del: 7*
Jour/Kursansvarig	*Per-Magnus Olsson*
Telefon under skrivtid	*5607*
Besöker salen ca kl.	*Varje hel timme*
Kursadministratör (namn + tfnnr + mailadress)	*Anna Grabska Eklund* *anna.grabska.eklund@liu.se*
Tillåtna hjälpmedel	*Teoretisk del: Inga* *Praktisk del: Den C++information som finns i systemet.*
Övrigt (exempel när resultat kan ses på webben, betygsgränser, visning, övriga salar tentan går i m.m.)	*Lösningsförslag kommer på kurshemsidan.*
Vilken typ av papper ska användas, rutigt eller linjerat	*Spelar ingen roll.*
Antal exemplar i påsen	*35*

Tentamen i TDP004
Objektorienterad Programmering
Teoretisk del

Datum:	2010-12-20
Tid:	14-18
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: 40 med 20 poäng vardera på teori och praktikdel.

Poäng	**Betyg**
35-40	**5**
28-34	**4**
21-27	**3**
0-20	**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 teoretisk uppgift på ett separat blad.

Upperolpgifterna är inte ordnade efter svårighetsgrad.

Lycka till!

TDP004 Objektorienterad Programmering
Teoretisk del

1. Se nedanstående klass

```
class Entity {

public:

   Entity(double x, double y, double z);

   ~Entity();

   /*Updates the position etc, deltaTime is the time

   since the last update*/

   void Update(double delta_time)

protected:

   int m_id;

   double m_speed

   double m_x;

   double m_y;

   double m_z;

 private:

   void Fight(Entity* opponent = m_old_opponent);

   Entity* m_old_opponent

};
```

Antag att klassen `Troll` ärver från `Entity`.

   a) Vilken tillgänglighet får funktionen `Update` och variabeln `m_id` om arvet är `public`? (2p)

   b) Vilken tillgänglighet får funktionen `Fight` och variabeln `m_id` om arvet är `private`? (2p)

2. I standardbiblioteket finns en mängd olika containrar av olika typer. Många, men inte alla, funktioner finns för flera containrar. Till exempel finns funktionen `push_back` för både `vector` och `list`, medan `push_front` finns till `list` men inte till `vector`. Motivera varför. För full poäng krävs en generell motivering som dels nämner den del av designfilosofin bakom STL som är relevant här, och dels ett exempel på varför `push_front` finns till `list`, men inte till `vector`. (3p)

3. Funktioner och variabler som är `static` är lite speciella. Beskriv vad som skiljer en `static` funktion från en vanlig klassfunktion samt vad som skiljer en `static` variabel från en vanlig lokal/ klass-variabel. (5p)

4. I C++ finns något som kallas `namespace`.

    a. Vad är fördelen med `namespace`? (2p)

    b. På vilka två sätt kan man göra för att använda en funktion som finns i ett visst `namespace`? (2p)

5. Antag att vi har en typ `T`. Ett objekt av typen `T` ska vara inparameter till en funktion, i vilken objektet inte får ändras. Hur skulle du göra parameteröverföringen i följande fall. Motivera!

    a. Om `T` är grundläggande typ, till exempel `int`? (2p)

    b. Om `T` är en mera komplicerad typ, t.ex. en klass som innehåller mycket data? (2p)