Försättsblad för skriftlig tentamen vid Linköpings Universitet

Sal	*SU15-16*
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: 6*
Jour/Kursansvarig	*Per-Magnus Olsson*
Telefon under skrivtid	*1456*
Besöker salen ca kl.	*Ungefär 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	*3*

Tentamen i TDP004
Objektorienterad Programmering
Teoretisk del

Datum:	2011-08-22
Tid:	14-18
Plats:	SU-salar i B-huset.
Jour:	Per-Magnus Olsson, tel 281456
	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: 42 med 21 poäng vardera på teori och praktikdel.

Poäng	**Betyg**
36-42	**5**
29-35	**4**
22-28	**3**
0-21	**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.
Uppgifterna är inte ordnade efter svårighetsgrad.
Lycka till!

TDP004 Objektorienterad Programmering
Teoretisk del

1) I vissa sorters korsord associeras varje bokstav med en siffra. I alla rutor där siffran finns ska den associerade bokstaven stå. Exempel: bokstaven B associeras med siffran 23, vilket betyder att i alla rutor i korsordet där siffran 23 står, ska bokstaven B skrivas. Naturligtvis får varje bokstav endast associeras med en siffra, och varje siffra får enbart associeras med en bokstav.

a) Vilken av de STL-containrar vi har gått igenom i kursen är lämplig för att spara ovanstående association? Ta endast hänsyn till följande fakta i den här deluppgiften samt informationen ovan: man lägger in ett begränsat antal associationer (= antalet bokstäver), antalet ”look up”/sökningar kan däremot bli mycket stort (beroende på hur duktig man är på att lösa korsord). Man vill alltid enbart ha reda på siffran som är associerad med en bokstav. Motivera ditt val av container (3p).

b) Antag nu att man även vill kunna hitta vilken bokstav som är associerad med en viss bokstav, alltså en omvänd ”look up” jämfört med a-delen (Exempel: vilken bokstav är associerad med siffran 23? Bokstaven B.). Vilken STL-container är lämplig nu? Den här containern ska naturligtvis även kunna hantera de relevanta kraven i a-delen. Motivera ditt val (3p).

2 a) När ett funktionsanrop sker och minne allokeras för lokala variabler, var allokeras det minnet? (2p)
b) När du skapar en `static`-variabel, var allokeras minnet för den variabeln? (2p)

3 En av orsakerna till att objektorientering utvecklades var återanvändning (eng. reuse) av kod. Beskriv hur detta underlättas inom objektorientering jämfört med icke objektorienterade språk. (2p)

4) a) Vad medför nyckelordet `friend`? (1p)
b) Det finns argument både för och emot varför `friend` bryter mot inkapsling. Ge ett argument för och ett emot (2p).

5) Du har följande kod, tagen från ett påhittat spel. Kodsnutten hanterar projektiler och kontrollerar huruvida de skapas inuti något föremål i omgivningen. Du kan anta att allt utom hanteringen av projektiler fungerar som avsett.

```
std::vector<Projectile*> projectiles;

projectile* p1 = new Projectile ();
```

```
   projectiles.push_back(p1);
            if(true == Environment.Intersection(p1))
            {
                        /* Error! Projectile created inside some object.
                        Delete projectile. */
                        delete p1;
            }
…
//Lots more things happen here.
…
//Program shutdown, remove all projectiles.
std::vector<Projectile*>::iterator i;
//Program crashes inside loop. WHY?
for(i = projectiles.begin(); i != projectiles.end(); i++)
{
            delete (*i);
}
```

Förklara varför programmet kraschar inuti loopen vid kommentaren och skriv kod för två möjliga olika sätt hur man kan lösa problemet. Förklara även varför dina ändringar löser problemet (6p).