# Tentamensregler

## Hjälpmedel

Följande får tas med på tentan:

- En bok om C++. För boken gäller följande regler:
  - Kommentarer eller noteringar som direkt rör text och exempel på sidan i fråga får finnas i sidmarginalen.
  - Egna sidflikar för att enkelt kunna hitta t.ex. de olika kapitlen är tillåtna.
  - Inga extra ark eller lappar, lösa eller fastsatta, får finnas.
  - Tomma sidor, insidan av pärmarna, försättsblad, etc., får inte innehålla programkod.
- Ett A4-ark med valfritt innehåll på vardera sidor (går ej att ersätta med två enkelsidiga ark).
- Penna för att anteckna under tentan. Ni kan be tentamensvakt om kladdpapper.

Följande får INTE tas med:

- Elektroniska prylar. Dit hör till exempel miniräknare, mobiltelefon, hörlurar, tangentbord, smartklocka etc.

## Utloggning

När du är nöjd med ditt betyg (som står i tentaklienten) kan du avsluta tentan. Stäng alla program och spara alla filer. Sedan loggar du ut som vanligt i menyn. Lämna inte din plats innan vanliga inloggningsskärmen syns.

## Frågor om uppgifter

Frågor om tentan i stort eller uppgiftspecifika frågor ska ställas via tenta-klienten. Detta för att vi ska ha en historik av konversationen samt för att vi ska kunna ge samma hjälp till olika studenter.

## Systemfrågor

Om du har systemfrågor som t.ex. problem med tentaklienten eller terminalen räcker du upp handen så kommer en assistent och hjälper till.

## C++ referens

Det finns tillgång till valda delar av `cppreference.com`. Du måste starta webbläsaren via skrivbordsikonen “Web access” för att komma åt sidan.

## Alias för kompilering

Under tentan finns det tre alias att använda sig av för kompilering med c++17:
`w++17` Rekommenderas!
`e++17` Kompilering med alla varningar som fel.
`g++17` Kompilering utan varningar.

# Bedömning

Tentamen har två delar. Du måste bli godkänd på del 1 innan inlämning på del 2 bedöms.

## Bedömning Del 1

Del 1 består av 4 områden med 1 fråga på varje. Detta är grundfrågor på kursinnehållet du helt enkelt ska kunna. Varje fråga är designad för att kunna besvaras inom 15 minuter även för den som läser och skriver i maklig takt. Områdena behandlar:

1. Absoluta grunder. Exempel: klass, objekt/instans, konstruktor, inkapsling, datamedlem, medlemsfunktion, public, private, ändringsbarhet.
2. Operatorer. Exempel: medlemsoperator, fri operator, C++-standard för operatorer, pre-inkrement, post-inkrement, jämförelseoperatorer, aritmetiska operatorer, tilldelningsoperatorer, inmatningsoperator, utmatningsoperator, (indexoperator, funktionsoperator).
3. Minneshantering. Exempel: statiskt minne, dynamiskt minne, allokering, avallokering, stack-minne, heap-minne, adress, referens, pekare, destruktor, speciella medlemsfunktioner, kopieringskonstruktor, kopieringstilldelning, flyttkonstruktor, flytt-tilldelning, minnesläcka, ägandeskap.
4. Klassrelationer. Exempel: arv, basklass, härledd klass, delegerande konstruktor, polymorfi, virtual, override, pure virtual, virtuell destruktor, abstrakt klass, skyddad medlem, UML, komposition, aggregation, association.

Frågorna på del 1 bedöms med antingen *Godkänt* eller *Ej godkänt*. Vid *Ej godkänt* kan du försöka igen. Du kommer *inte* få några tips om vad som behöver åtgärdas. Uppgiften är så liten att du väntas läsa frågan igen för att uppfylla allt som står enligt god konvention. Om du försöker flera gånger utan konkreta framsteg sätts *Underkänt*. Du är då underkänd på tentamen och kan gå hem så snart tentamensvakten tillåter att du lämnar salen.

## Bonus på del 1

Varje laboration i kursen har en extrauppgift. Genom att lösa den tillgodoräknas du motsvarande uppgift i del 1:

| Uppgift | Tillgodo från |
|---|---|
| 1 | Måste alltid lösas på tentan |
| 2 | Tillgodo från Klockslagets extrauppgift |
| 3 | Tillgodo från Listans extrauppgift |
| 4 | Tillgodo från Pacmans extrauppgift |

## Bedömning Del 2

Del 2 på tentamen består av två större uppgifter. Här visar du att du kan kombinera flera av kunskaperna från kursen för att lösa ett givet problem. Varje uppgift bedöms med upp till tre poäng:

3p För att en uppgift ska anses godkänd för 3 poäng krävs följande:

- Uppgiften är fullständigt löst enligt givna instruktioner.
- Alla krav uppfylls i generella fallet (körexempel är bara enstaka exempel).
- Koden följer konsekvent och tydlig stil.
- Koden följer god C++-konvention (t.ex. inga kompileringsvarningar).
- Klasser har ett tydligt ansvar och funktioner en väl avgränsad uppgift.
- Klasser är inkapslade med god resurshantering och felhantering.
- Det finns inga övriga brister att tala om.

2p För två poäng är punkterna ett och två för 3 poäng uppfyllda, och bland övriga punkter är det bara *en* som inte uppfylls.

1p Används inte.

0p *Två eller fler* av kraven för 3p är inte uppfyllda.

I del 2 kommer du att få återkoppling på din inlämning som ger dig viss ledning till vad som behöver åtgärdas för att uppnå nästa poängnivå. När du löst problemet kan du skicka in igen. Skickar du in flera gånger utan konkreta framsteg *fryser* vi uppgiften på uppnådd poängnivå. Du kan då inte försöka mer på den uppgiften.

För respektive betyg i kursen krävs lika många poäng:

| Betyg | Poäng |
|---|---|
| 3 | 3 |
| 4 | 4 |
| 5 | 5 |

## Bonus på del 2

Genom visat engagemang i kursen (närvaro på lektioner, ligga i fas, noggrannhet vid kompletteringar) kan du få 1p extra på del 2. Denna bonus förs in under **Uppgift 7** när du uppnått minst 2 poäng.
Löser du en uppgift med 2p eller två uppgifter med 2p är bonusen skillnaden mellan betyg U och 3 eller betyg 4 och 5 respektive.

# Del 1: måste lösas före del 2

## Uppgift 1

Skapa en klass för att representera ett heltal inom intervallet 15 till 35. När ett objekt av klassen skapas ska ett heltal kunna anges. Givet ett objekt ska man kunna hämta ut heltalet ur objektet eller ändra heltalets värde. Ett befintligt objekt ska garantera att dess heltal alltid är inom ovan angivet intervall.

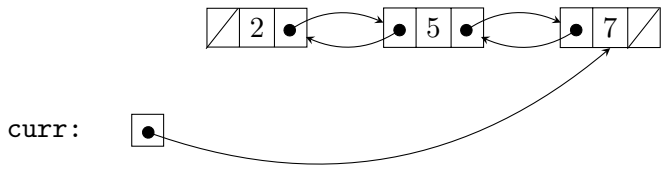
Denna uppgift kräver inga operatorer eller speciella medlemsfunktioner och all kod kan skrivas i filen `uppgift1.cc`.

## Uppgift 2

Implementera en operator som multiplicerar en sträng med ett heltal. Resultatet ska vara en sträng där strängparametern har upprepats det antal gånger som heltalsparametern anger.

**TIPS:** std::string har redan en `operator+=` som eventuellt kan användas.

## Uppgift 3




Du har ovanstående struktur för en lista. Skriv kod som tar bort elementet med värde 5 ur listan. Det du har tillgång till är variabeln `curr`. När du är klar ska `curr` fortfarande peka på elementet med värde 7. Antag noder av typ `struct Node{ Node* prev; int val; Node* next; }`

**KRAV:** Du får inte byta plats på värden, endast flytta pekare. Samtliga element är dynamiskt allokerade och koden får inte generera minnesläckor.

## Uppgift 4

Du har klasserna A och B. Klassen A består av en sträng som representerar en färg, ett heltal som representerar ett avstånd och ett flyttal som representerar en styrka. Klassen B består av ett heltal som representerar ett avstånd och ett flyttal som representerar en volym. Både klassen A och B implementera en version av funktionen `get_distance()`. A har en funktion `get_color()` som returnerar dess färg och B har en funktion `get_color()` som alltid returnerar färgen “grå”. Dessutom behöver varje klass en funktion för att öka sin styrka respektive volym.

Du kan utgå från att både A och B är en C. Din uppgift är att implementera en lämplig klassdefinition för basklass C (dvs det som brukar finnas i en headerfil).

# Del 2: bedöms först när del 1 är godkänd

## Uppgift 5

Det finns en uppsjö så kallade “merge”-spel. Det som kännetecknar ett sådant är att spelaren erbjuds ett antal rutor med objekt som kan dras runt mellan rutorna. Om ett objekt släpps på en ruta med ett exakt likadant objekt så uppgraderas objektet på destinationsrutan en nivå medan det släppta objektet försvinner från sin ruta.

Den inte helt okända men relativt nya spelstudion Oskho Games ska försöka slå sig in på marknaden med ett nytt merge-spel. Du har anlitats för att implementera grundmekaniken i spelet. Den består av klassen `Inventory_Space` som representerar en ruta i spelet. Rutan kan vara antingen tom eller peka ut ett dynamiskt allokerat objekt av typen `int`.

Exempel på huvudprogram finns i `uppgift5.cc`.

Din klass `Inventory_Space` ska implementera:

- Lämplig konstruktor för att givet huvudprogram ska fungera.
- `A.print()` skriver ut innehållet på ruta A, eller ett bindestreck om ruta A är tom.
- `A.merge(B)` släpper innehållet i ruta B på ruta A.
  - Om rutan släpps på sig själv ska inget hända (dvs när ruta A och ruta B är samma ruta).
  - Om ruta A är tom ska ruta B *flyttas* till ruta A.
  - Om ruta A och ruta B har samma värde ska värdet i ruta A ökas med 1 och innehållet i rutan B tas bort.
  - I alla andra fall ska inget hända.
- Speciella medlemsfunktionerna kopieringskonstruktorn, flyttkonstruktorn samt destruktorn. Flytt-tilldelning och kopieringstilldelning ska explicit tas bort.
- Korrekt minneshantering. Inga minnesläckor tillåts.

Det givna huvudprogrammet kommer vid korrekt implementation skriva ut:

```
2111111-
221111--
31111---
3211----
322-----
33------
4-------
4-------
```

## Uppgift 6

I denna uppgift ska du skapa klasser som beskriver en varukorg. En varukorg består av ett antal produkter. Det ska gå att köpa valfritt antal av varje produkt. Varje produkt består i sin tur av ett namn och ett styckpris.

Skapa en klass `Product` för att representera en produkt. Du behöver kunna be produkten om dess pris och skriva ut produkten på valfri utström. Utskrift ska ske genom en operator.

Skapa därefter en klass `Shopping_Cart` som innehåller en `vector` av produkter. Du behöver också hålla reda på antal av respektive produkt. Klassen ska ha funktionalitet för att lägga till en produkt (ned tillhörande antal), beräkna totala varuvärdet och skriva ut hela varukorgen på en given valfri utström. Utskrift ska följa exempel nedan. Utskrift ska ske genom en operator.

Exempel på huvudprogram finns i filen `uppgift6.cc`.

```
apelsin: 2st * 5.00kr
banan: 5st * 7.50kr
citron: 1st * 15.00kr
Total: 62.50kr

gurka: 1st * 9.00kr
tomat: 5st * 18.50kr
sallad: 1st * 11.00kr
Total: 112.50kr

smör: 1st * 19.00kr
bröd: 10st * 24.50kr
ost: 1st * 48.00kr
Total: 312.00kr
```

**KRAV:** Du ska skriva din kod med korrekt filuppdening.

**TIPS:** Det finns flera sätt att hålla koll på kopplingen mellan produkt och antal. Ett av dessa är att skapa en strukt med produkt och antal.