Regler

Regler - generellt

- Frågor ställs genom Zoom. Klicka på knappen (Ask for Help) så hjälper vi dig så fort som möjligt.
- Slutinlämningen av din kod ska vara skriven enligt god programmeringssed för C++. Det betyder att tekniker som finns för att underlätta användning, utveckling, felsökning och underhåll ska användas.
- Du ska sitta i en ostörd miljö utan andra personer i samma rum. Du ska hela tiden vara uppkopplad och synlig i Zoom. Om du är klar med en deluppgift och har lämnat in den kan du sätta en lapp framför din kamera där det står "Rast". Om du är klar med tentan kan du logga ut.
- Du kommer behöva legitimera dig under tentamen. Ha ditt foto-id redo.
- Var beredd på att i efterhand kunna redogöra för dina svar.
- Om du behöver ta en rast, lämna in de filer du jobbar på just då till "2020-08-25: Rast" i Lisam.
- Alla former av regelbrott medför att din tentamen underkänns direkt.
- Information och dina givna filer kommer att publiceras under tentans gång på följande sida: `https://www.ida.liu.se/~TDDI82/exam/2020/2020-08-25/index.sv.shtml`

Regler - Tider

- STL-uppgiften publiceras 14:30 och lämnas senast in 16:00.
- 16:00 till 16:15 är det rast.
- Mall-uppgiften publiceras 16:15 och lämnas senast in 17:45.

Regler - hjälpmedel

- All kommunikation är förbjuden, undantaget frågor till kurspersonal.
- All form av kopiering eller avskrift är förbjuden.
- Alla källor du tar inspiration av ska redovisas.
- Du får använda `cppreference.com` fritt.
- Du får använda en valfri C++ bok
- Du får använda en A4 sida med anteckningar. Du måste lämna in dina anteckningar i inlämningen "2020-08-25: Anteckningar" i Lisam innan du börjar skriva tentan (går att lämna in fr.o.m kl 08:00).

Regler - betyg

Tentan består av två uppgifter, en för STL och en för Mallar. Varje del bedöms som U, G eller VG. För att få G på en uppgift måste du ha uppfyllt *majoriteten* av kraven som presenteras i uppgiften. Du måste även ha gjort ett försök till att svara på alla frågor i uppgiften. Detta betyder alltså att du kan få godkänt även om du inte har en fullt fungerande lösning.

I tabellen nedan presenteras betygsgränserna:

Betyg	STL	Mallar
3	G	G
4	G	VG
4	VG	G
5	VG	VG

Det krävs också att du fyllt i tentans regelinlämning, vilket bekräftar att du läst tentans regler och tänker följa dem. Bonus insamlad under kursen gäller inte denna tenta. All bonus behåller du till dess att nästa ordinarie datortenta kan genomföras.

Regler - inlämning

Inlämning görs via Lisam, på följande sida: `https://studentsubmissions.app.cloud.it.liu.se/Courses/TDDI82_2020VT_CF/`. Du kan även hitta sidan genom att gå in på TDDI82 i `https://lisam.liu.se` och sedan klicka på “Inlämningar” i menyn.

Följande inlämningar kommer vara tillgängliga vid de specificerade tiderna:

- 2020-08-25: Regler (14:00-17:45)
- 2020-08-25: Anteckningar (14:00-15:00)
- 2020-08-25: Rast (14:30-17:45)
- 2020-08-25: STL Kod (14:30-16:00)
- 2020-08-25: STL PDF (14:30-16:00)
- 2020-08-25: Mallar Kod (16:15-17:45)
- 2020-08-25: Mallar PDF (16:15-17:45)

I “2020-08-25: STL Kod” lämnar du in koden till din lösning för STL-uppgiften och i “2020-08-25: STL PDF” lämnar du in dina svar på frågorna till STL uppgiften som en PDF fil. Likadant för Mall-uppgiften.

Regelinlämning

Innan du börjar arbeta på första uppgiften måste du göra en inskickning till “2020-08-25: Regler” i Lisam (se ovan).

Du kan skicka ett meddelande via Lisam där du bekräftar att du har läst reglerna och att du tänker följa dem.

Kom ihåg att om du har en sida med anteckningar måste du skicka in denna i “2020-08-25: Anteckningar” i Lisam. Om det är datorskrivet kan du skicka in dem som de är. Om de är handskrivna kan du antingen skanna in dem eller ta en bild.

Gör detta innan du börjar jobba på tentan!

Mallar

Tidsplan

Vi rekommenderar att du lägger ungefär 60 minuter på att skriva kod och 30 minuter på att svara på frågorna. Om du får ont om tid är det bättre att du skriver bra svar på frågorna än att du lägger tid på att göra klart koden.

Uppgift

I den givna filen `mallar.cc` implementeras två klasser: `List_Policy` och `Printer`.

Printer är en klass som definierar ett sätt att skriva ut en databehållare på (i dagsläget funkar den endast för `std::vector<int>`).

List_Policy Innehåller den funktionalitet och data som krävs för att skriva ut en databehållare som en lista.

Idén är att i slutändan ska man kunna variera exakt hur `Printer` skriver ut databehållare genom att byta ut `List_Policy`. För att uppnå detta måste vi slutföra vissa delmål. Dessa är:

1. `Printer` och `List_Policy` måste fungera för olika datatyper, inte bara `int`.
2. Det måste skapas två versioner av `Printer::print`, en som tar två stycken *godtyckliga iteratorer* och en som tar en *godtycklig databehållare*. För tillfället finns det endast en version av `print`, som tar emot två stycken `std::vector<int>::iterator`. Denna version kan användas som en bas för den version som tar godtyckliga iteratorer. Notera att varken iteratorerna eller databehållaren är kopplade till `Printer` klassen, utan det är funktionen `print` som ska hantera detta.
3. `List_Policy` måste gå att byta mot en godtycklig klass som innehåller de två funktionerna `print` och `done`.

För att bli **godkänd** på uppgiften måste du lösa punkt 1 och punkt 2. D.v.s. du måste:

- Skriva om `List_Policy` och `Printer` så att de hanterar godtyckliga datatyper istället för `int`.
- Skriva om `Printer::print` så att den tar emot godtyckliga iteratorer (det ska alltså vara en mallifierad medlemsfunktion).
- Lägga till en överlagring av `Printer::print` som tar emot en godtycklig databehållare. **Tips:** inuti i denna överlagring kan du anropa den iterator-baserade överlagringen.

Testa din lösning genom att lägga till eller ändra testerna i `main` (du kan återanvända de givna testerna för olika datatyper).

Notera att det finns frågor under VG delen som du måste svara på för godkänt.

VG

För att få VG på denna uppgift måste du, efter att du har löst kraven ovanför, lösa punkt 3 genom att lägga till ytterligare en mallifierad klass vid namn `Table_Policy` som måste uppfylla följande krav:

- Tar exakt en mall parameter `T` som är en godtycklig datatyp.
- Har en konstruktor som tar emot två stycken heltal: `width` och `columns`.
- Innehåller en mallifierad medlemsfunktion `print` som tar emot två parametrar: en utström och en referens till det `T` objekt som ska skrivas ut.
- Har en funktion `done` som tar emot en utström.

Syftet med denna klass är att, istället för att skriva ut en databehållare som en lista så skrivs den ut som en tabell. `width` avgör hur bred (i antalet tecken) varje cell i tabellen är och `columns` bestämmer hur många kolumner tabellen innehåller.

För tips på hur du kan implementera `print` och `done`, se den utkommenterade koden i `mallar.cc`.

Utöver att implementera `Table_Policy` så måste du även utöka `Printer` så att den tillåter användaren att välja antingen `List_Policy` eller `Table_Policy` för att skriva ut databehållaren. Detta innebär alltså att `policy` måste vara en godtycklig datatyp istället för `List_Policy`.

Frågor

Svara på ***ALLA*** frågor nedan angående din lösning, både för G och för VG. Skriv dina svar i ett separat PDF dokument. Detta dokument ska som mest vara 1000 ord långt.

1. Vad tycker du är bra, respektive dåligt i designen av `Printer` i din lösning? Du kan fundera på (men är inte begränsad till) läslighet, användbarhet och skalbarhet (hur lätt det är att utöka funktionaliteten).
2. Måste `Printer` ta emot en godtycklig datatyp? Förklara.
3. Kan du identifiera några operationer som används i `Printer`, `List_Policy` eller `Table_Policy` som måste fungerar med den mallifierade datatypen. Om du kan det, förklara varför dessa behövs. Om inte, förklara varför det inte finns några begränsningar.

Svara på frågorna ovan så gott du kan. Vi letar inte nödvändigtvis efter ett specifikt svar utan vi vill se din tankegång.