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".
- 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 "2020-06-01: 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/2020/2020-06-01/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 "2020-06-01: Anteckningar" i Lisam innan du börjar skriva tentan.

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:

- 2020-06-01: Regler (14:00-17:45)
- 2020-06-01: Anteckningar (14:00-15:00)
- 2020-06-01: Rast (14:30-17:45)
- 2020-06-01: STL Kod (14:30-16:00)
- 2020-06-01: STL PDF (14:30-16:00)
- 2020-06-01: Mallar Kod (16:15-17:45)
- 2020-06-01: Mallar PDF (16:15-17:45)

I “2020-06-01: STL Kod” lämnar du in koden till din lösning för STL-uppgiften och i “2020-06-01: 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 “2020-06-01: 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 “2020-06-01: 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` finns det två klasser definierade. Dessa är `Sum_Handler` och `Accumulator`.

Accumulator är en *stack* som tillåter användaren att slå ihop hela innehållet på stacken m.h.a. funktionen `evaluate`.

Sum_Handler Innehåller de funktioner och den data som behövs för att summera innehållet i `Accumulator`.

Tanken är att olika instanser av `Accumulator` kan variera hur innehållet slås ihop genom att ange en specifik `Handler`. För tillfället är dock `Accumulator` väldigt begränsad.

1. Den kan bara innehålla `int`
2. Det går bara att initialisera värdena från `std::vector` iteratorer
3. Den kan bara använda `Sum_Handler` för att slå ihop stacken

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

- Skriva om `Accumulator` och `Sum_Handler` så att de hanterar godtyckliga datatyper istället för `int`. Du får anta att dessa godtyckliga datatyper har `operator+`.
- Skriva om konstruktorn till `Accumulator` så att den kan initieras med två stycken iteratorer från en godtycklig databehållare istället för iteratorer från `std::vector`.

Testa din lösning genom att lägga till eller ändra testerna i `main`.

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 klass vid namn `String_Handler` som måste uppfylla följande krav:

- Den ska ha en funktion `combine` som tar två parametrar, en sträng och ett värde av en godtycklig datatyp `T`. Denna funktion kombinerar en `std::string` vid namn `result` med ett värde av datatypen `T` och returnerar resultatet som en `std::string`. Detta gör den genom att konvertera `T`-värdet till en sträng och lägger till den i slutet av `result` (kan vara bra att lägga till ett mellanslag efteråt så att värdena separeras).

- Den ska ha en funktion `initial` som returnerar en tom sträng.
- Den ska inehålla ett typalias av `std::string` vid namn `value_type`.

Tips: Strängströmmar tillåter dig att enkelt slå ihop en sträng med ett värde, givet att `T` har `operator<<` (vilket du får anta att den har).

Utöver att skapa `String_Handler` så måste du även lägga till ytterligare en mallparameter till `Accumulator` som tillåter användaren att välja huruvida stacken ska kombineras m.h.a. `Sum_Handler` eller den nyligen skapade `String_Handler`. Detta innebär alltså att `handler` kan vara en godtycklig datatyp istället för `Sum_Handler`.

Tips: Fundera noga på vad `Accumulator::evaluate` måste ha för returtyp för att få detta att fungera med både `Sum_Handler` och `String_Handler`.

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 finns det för fördel med att låta en annan klass bestämma hur `Accumulator` ska slå ihop alla värden?
2. Varför sparas `handler` som en datamedlem i `Accumulator`?
3. Hur skiljer sig filuppdelning (`.h` och `.cc` filer) när man jobbar med klassmallar jämfört med vanliga klasser?

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