TDDI82 – Tentaregler

Hjälpmedel

Följande får tas med på tentan:

- En bok om c++. För boken gäller följande regler:
  - Kommentarer/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.
- Maximalt ett A4-ark med egna anteckningar
- Penna för att anteckna under tentan. Ni kommer förses med blanka papper

Följande får INTE tas med:

- Elektroniska hjälpmedel såsom miniräknare, mobiltelefon och smartklockor.

Utloggning

När du är nöjd med ditt betyg (som står i tentaklienten) kan du avsluta och logga ut som vanligt i menyn. Klicka sedan på ok följt av knappen avsluta tentan. Observera att du när du gjort detta inte kan logga in igen. 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.

Tentaregler

Tentan består av fyra uppgifter indelade i två kategorier; standardbibliotek (STL) och mallar. För godkänt betyg på tentan krävs godkänd lösning av en uppgift inom vardera kategori. För högre betyg krävs lösning av uppgifter inom en viss tid enligt tabell 1.

Tid (h)	Antal uppgifter		Betyg
	STL	Mallar	
2.5 + B	1	1	5
4 + B	2	1	5
4 + B	1	2	5
4 + B	1	1	4
5	1	1	3

Tabell 1: Tidsgränser för olika slutbetyg, B är bonus från labserien (se nedan)

För att en uppgift ska anses godkänd krävs följande:

- att man noga följt alla instruktioner och krav ställda i uppgiften
- din kod följer god programmeringsstil (se labseriens rättningsguide)
- att din kod har bra inkapsling och resurshantering
- att standardbibliotekskomponenter används på ett bra sätt

Bonus från labserien

Bonus från labserien (benämnd B i tabell 1) ger ett visst antal minuter (5, 15, 20 eller 30) extra på respektive tidsgräns för högre betyg. Bonusen är endast giltig det år den erhölls.

C++ referens

Det finns tillgång till valda delar av `cppreference.com`. Du måste starta webbläsaren via menyn för att komma åt sidan.

Alias för kompilering

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

Testning med catch

Vissa uppgifter använder sig av catch2. Kopiera filerna `catch.hpp` och `test_main.cc` från given_files till din katalog och kompilera huvudprogrammet separat: `g++17 -c test_main.cc`
Du får då en o-fil som kan användas för att kompilera dina testfall: `w++17 fil.cpp test_main.o`

Uppgift 1 - STL

I denna uppgift är det extra viktigt att du använder standardbibliotekets algoritmer i så stor grad som möjligt. Egna upprepningssatser (`for`, `while` eller `do`) tillåts inte.

I denna uppgift ska du skapa ett program som behandlar *anagram*.
Två ord är *anagram* av varandra om de består av *exakt* samma bokstäver, men i olika ordning.
Exempel: "anagram" och "magarna" är *anagram* av varandra.

I `given_files/anagram.cc` finns det en tom funktion `are_anagrams` som tar två strängar. Du ska implementera funktionen `are_anagrams` så att den returnerar `true` om de inskickade strängarna är anagram av varandra och `false` annars.
Ditt huvudprogram ska göra följande (exklusive utskrifter):

1. Läs in ett ord från `cin` till en variabel `word`
2. Läs in flera ord till en vektor `text` tills den når filslut (`ctrl-D`)
3. Ta bort alla ord i `text`-vektorn som är anagram till `word` (använd `are_anagrams`)
4. Skriv ut innehållet av `text` separerade med mellanslag

Utskrifterna ska följa körexempel nedan.

Körexempel (fetstilt är användarinmatning)

```
Mata in ett ord: eskort
Mata in en text: etisk sektor med en eskort skoter maskin
etisk med en maskin

Mata in ett ord: avdrag
Mata in en text: garvad i sin vardag om tid avdrag
i sin om tid
```

Uppgift 2 - STL

Du har en fil, `given_files/names.txt`, där varje rad innehåller ett namn, ett kolon och flera mellanslagsseparerade taggar (ord). Din uppgift är att skapa ett program som låter användaren mata in ett antal taggar och sedan skriver ut alla namn som har samtliga taggar. Detta ska göras så långt som möjligt med hjälp av standardbibliotekets containrar och algoritmer enligt nedanstående algoritm:

1. Öppna filen för läsning
2. Skapa en map för att koppla en tagg till alla namn som har den taggen (alltså rekommenderas typen `map<string, vector<string>>`) kallad `tags`. Observera att taggen alltså är nyckeln.
3. För varje rad i filen, läs in namnet och lägg till det i vectorn som hör samman med varje tagg på raden.
4. Sortera alla namn-vektorer
5. Låt användaren mata in taggar, spara dem i en vector
6. Skapa en ny `vector<string>`, `common_names` och initiera den med namnen som hör samman med den första inmatade taggen
7. För alla inmatade taggar utom den första, skapa en ny `vector<string>` där du sparar de namn som förekommer både i `common_names` och bland namnen som hör till aktuell tagg. Gör detta med hjälp av `set_intersection`. Skriv sedan över `common_names` med namnen i den nya vectorn.
8. Skriv ut namnen i `common_names`, ett namn per rad.

Körexempel (fetstilt är användarinmatning)

```
Mata in taggar: MEXICO SWEDEN
Gemensamma namn
SOFIA

Mata in taggar: SWEDEN NORWAY ICELAND FEMALE
Gemensamma namn
ANNA
EMMA
SARA
```

Uppgift 3 - Mallar

Din uppgift är att skapa en generell mallfunktion `sum` för att summera värden. `sum` tar ett argument som antas vara någon typ av container ur standardbiblioteket och funktionen ska summera elementen i containern och returnera denna summa. Du kan anta att elementtypen stödjer `operator=`, `operator+` och kan default-initieras (har en default-konstruktor).

Krav

Följande krav måste beaktas i denna uppgift:

1. Samtliga STL-containrar (där `value_type` har stöd för de angivna operationerna) ska stödjas (även de som endast stödjer ForwardIterator)
2. Returtypen ska vara samma som elementtypen

På filen `given_files/sum.cc` finns det några testfall att testa din funktion med. Observera att du kan behöva lägga till testfall för att vara säker på att funktionen fungerar enligt ovan.

Uppgift 4 - Mallar

I denna uppgift ska du skapa en klassmall `Cycle` som:

- tar en mallparameter `Container` och sparar en referens till en (random-access) databehållare av godtycklig typ (bestämms via `Container`). Kalla denna behållare för `container`.
- har en lämplig konstruktor som initierar `container`.
- innehåller `value_type` vilket är ett typalias för samma typ som elementen i `container`.
- har en funktion `size` som returnerar resultatet av `container.size()`.
- har en `at`-funktion som tar ett index och returnerar en referens till motsvarande element i `container`. Om detta index är negativt ska ett undantag av lämplig typ kastas.
  Om index är lika med `container.size()` ska `at` returnera en referens till element `0`; om index är lika med `container.size() + 1` returneras en referens till element `1` o.s.v.
  D.v.s. `at` ska agera cykliskt och ska alltid returnera en referens till ett lämpligt element i `container` för positiva värden.
- Det ska finnas två versioner av `at`, en `const`-variant och en icke-`const`-variant.

Det finns givna testfall i `given_files/cycle.cc` som alla ska gå igenom när du har implementerat `Cycle`. Du kan behöva lägga till egna testfall för att försäkra dig om att allting är korrekt.