# TDDI82 – Tentaregler

## Inloggning

Logga in i tentasystemet genom att välja session ”exam system” och logga in med ditt vanliga LiU-ID. Välj **inte** att ha denna session som standardsession. Verifiera att dina uppgifter stämmer och förbered din tentaplats. Som vanligt är det inte tillåtet att ha väskor eller jackor vid sin skrivplats och mobiltelefoner ska ligga avstängda i jacka eller väska. Ta fram ditt LiU-kort och invänta tentavakt för att få ett engångslösenord.

## 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 - Mallar

I denna uppgift ska du skapa en mallfunktion `reverse` som tar emot en container som åtminstone har *Bidirectional*-iteratorer och returnerar en likadan container med samma värden men i omvänd ordning. I filen `given_files/reverse.cpp` finns det testfall som visar på hur det är tänkt att det ska fungera.

Du behöver inte kontrollera att containern uppfyller kraven på Bidirectional-iterator utan det kan du anta. Du kan även anta att elementtypen stödjer både kopiering och flytt.

# Uppgift 2 - STL

I denna uppgift är det extra viktigt att du använder lämpliga behållare från standardbiblioteket för att förenkla din kod.

XKCD är en tecknad serie av Randall Munroe, före detta entreprenör för NASA, som enligt författaren själv handlar om ”romantik, sarkasm, matematik och språk”.[1] Seriestripparna 1288, 1625 och 1679 handlar alla om att systematiskt byta ut vissa ord i nyheterna “that make reading the news more fun”.[2] Filen `given_files/SUBSTITUTIONS.TXT` innehåller utbyten från de tre seriestripparna. Formatet är `SÖKORD:ORD ATT BYTA MED` där då SÖKORD ska bytas ut med ORD ATT BYTA MED (som kan vara fler än ett). Allt som återstår för dig är att skriva ett program som läser in en fil med föreslaga utbyten och sedan byter ut förekomster av sökorden i en text användaren matar in. Alla ord ska jämföras med gemener (små bokstäver).

KRAV: Du får inte sätta in ord från den inlästa texten i någon behållare (utom i `std::string` förstås). Eftertänksam användning av `const` kan hjälpa dig undvika misstag med detta.

### Körexempel 1 (Användarinmatning i fet stil)

```
$ a.out
Scientists disrupt space around car smartphone to win debate in senator election

channing tatum and his friends destroy spaaace around cat pokédex to win dance-off in elf-lord eating contest
```

### Körexempel 2

```
$ a.out < given_files/TEXT.TXT
Programutskriften utelämnas av utrymmesskäl.
```

[1] https://sv.wikipedia.org/wiki/Xkcd
[2] https://www.xkcd.com/1288/

# Uppgift 3 - 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`) ska inte behövas.

Du ska skapa ett program som skriver ut ett tal med alla dess primtalsfaktorer. Detta ska lösas med följande algoritm:

1. Skapa en vector med plats för 10 heltal.
2. Fyll vectorn med slumptal i intervallet $[2, 75]$
3. Använd den givna funktionen `is_prime` i filen `given_files/primtal.cpp` för att ta bort de tal som inte är primtal.
4. sortera vectorn
5. Beräkna produkten av talen i vectorn
6. Skriv ut produkten (det genererade talet) samt de värden som är kvar i vectorn (primtalsfaktorerna).

### Körexempel 1 (Resultatet kan så klart vara ett annat)

```
Random number: 3887
factors: 13 13 23
```

# Uppgift 4 - Mallar

I denna uppgift ska du skapa en funktionsobjektsklass, `TablePrinter` (klass med funktionsanropsoperatorn, `operator()` överlagrad) som kan användas för att skriva ut värden i tabellformat.
När man skapar ett objekt av typen `TablePrinter` ska användaren ange en ström där värden ska skrivas ut, bredden för varje värde samt hur många kolumner tabellen ska ha. Nedanstående exempel ska skriva ut en tabell med fem kolumner där varje värde tar åtta tecken:

```
vector vals {1,2,3,4,5,6,7,8,9,10,11,12,13};
for_each(begin(vals), end(vals), TablePrinter{cout, 8, 5});
```

Utskriften (på cout) skulle därmed vara denna:

```
       1       2       3       4       5
       6       7       8       9      10
      11      12      13
```

För att göra detta generellt (så att vilken typ som helst ska kunna skrivas ut) ska du använda templates. Det finns två sätt att lösa detta på, antingen är klassen en template eller så gör du själva funktionsanropsoperatorn till en mallfunktion. Det är upp till dig att välja vilken av modellerna du vill använda. Väljer du att göra klassen till en mall behöver du lägga till templateparametern i anropet i exemplet.