TDIU20 – 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.
- Ett A4-ark med valfritt innehåll på vardera sidor (går ej att ersätta med två enkelsidiga ark).
- Penna för att anteckna under tentan. Ni kommer förses med blanka papper

Följande får INTE tas med:

- Elektroniska hjälpmedel såsom miniräknare och mobiltelefon

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.

Tentaregler

Tentan har fyra praktiska uppgifter och en teoretisk uppgift. Två praktiska uppgifter är på grundnivå. Där visar du att du uppnår kraven för godkänt. De två praktiska uppgifterna och den teoretiska uppgiften är på djupare nivå. I dessa kan du visa att du nått upp till kraven för ett högre betyg. I den teoretiska uppgiften visar du att du kan använda rätt terminologi och förstått syften som behöver förklaras i text. Bedömning av den teoretiska uppgiften sker efter tentans slut. Testa dig själv genom att resonera kring instuderingsfrågorna.

För betyg 3 på tentan krävs lösning av en uppgift på grundnivå. För betyg 4 krävs dessutom lösning av en påbyggnadsuppgift och för betyg 5 krävs grunduppgift och två påbyggnadsuppgifter.
För att en uppgift ska anses godkänd krävs följande:

- att du noga följt alla instruktioner och krav ställda i uppgiften
- din kod följer god programmeringsstil (se labseriens rättningsguide)
- att klasser har ett tydligt ansvar och funktioner har en väl definierad uppgift
- att din kod har bra inkapsling och resurshantering

Bonus från labserien

Om du har bonus från labserien minskas antalet påbyggnadsuppgifter som krävs med ett, dvs för betyg 4 krävs endast en grunduppgift och för betyg 5 krävs en av varje. Bonusen är endast giltig det år den erhölls.

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 skrivborsikonen "Web access" för att komma åt sidan.

Alias för kompilering

Det finns 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.

Uppgift 1 - Grundnivå

I programspråket Python går det att enkelt och kortfattat hantera text. Som exempel visas koden för att läsa in en rad och sedan skriva ut orden i omvänd ordning:

```
line = input().split()
for word in reversed( line ):
  print(word)
```

I C++ blir det krångligare att göra samma sak. I `uppgift1.cc` finns ett kodexempel hur det kan se ut. Där finns även tre kodexempel som visar hur vi skulle vilja skriva koden. Du ska skapa en väl inkapslad klass i C++ för att göra detta möjligt. Här listas det första exemplet du ska få att fungera:

```
Word_String line;
cin >> line;

for ( int i{line.length()-1}; i >= 0; --i )
{
  std::cout << line.at(i) << std::endl;
}
```

Du ska alltså skriva klassen `Word_String`. Den ska ha en operator för att läsa in en rad text samt ett gränssnitt för att kunna indexera fram varje ord i raden. Gränssnittet består i grunden av funktionerna `at()` som returnerar N:te ordet på raden och `length()` som returnerar antalet ord på raden. Internt läses orden in med hjälp av strängströmmar och lagras i en `std::vector`. Du får lägga till namngivna funktioner i det publika gränssnittet som motsvarar respektive operator. Du får lägga till valfria privata hjälpfunktioner för att undvika koduppprepning.

- Vi bedömer hur väl din lösning fungerar med en omodifierad version av `uppgift1.cc`.
- Vi bedömer din lösning utefter hur få kompileringsvarningar den får med `w++17`.
- Skriv lösningen med filuppdelning. Klasser som hör intimt samman får ligga i samma fil.
- Sammansatta typer ska parameteröverföras utan kopiering och utan risk för oavsiktlig ändring.
- Medlemsfunktioner ska vara användbara på konstanta objekt i så stor utsträckning som möjligt.
- Konvertering från rad till vektorinnehåll ska lösas utan koduppprepning.
- Dina klasser ska vara väl inkapslade och följa god konvention. Det finns inget behov att använda "set"-funktioner eller vänfunktioner.

Det givna huvudprogrammet ska vid inmatning av fetstil text ge följande utmatningar:

Körexempel 1

```
This line is keyboard input
input keyboard is line This
This line too
too line This
This string constructor should also work
Finally you must add the index operator []
```

Uppgift 2 - Grundnivå

Bolaget Allmänna Järnvägar Sverige (AJS)[1] är en tills nu okänd konkurrent till mer välkända Statens Järnvägar Aktiebolag (SJAB)

För att bättre kunna spåra förseningar till följd av så kallade signalfel och andra begränsningar bygger AJS ett nytt system för att övervaka sina spår. I det nya systemet är alla järnvägsspår indelade i `Segment`. Ett segment är ett enkeltriktat enkelspår utan några förgreningar med en viss längd i kilometer. På varje segment kan det finnas noll eller flera `Train`. Varje tåg har ett antal vagnar och kör med en viss marschfart. Både tåg och segment ska kunna namnges.

Maximalt tillåtna högsta hastighet på vilket spår som helst är 300 km/h.

Du ska implementera klasser för att representera segment och tåg. Av säkerhetsskäl får det i snitt inte vara mer än 10 vagnar per kilometer på ett givet segment. Tåg kan alltså behöva vänta om ett segment av ovan skäl är fullt. Hur tåg väntar lösas av den som använder dina klasser så det behöver du inte bry dig om.

Den som ska använda klasserna önskar att det är enkelt att kontrollera om ett tåg får åka in på ett segment och dessutom enkelt att kontrollera att tåget kan bibehålla önskad hastighet utan att åka in i tåg som är framför. Hen har skrivit ett exempel på hur hen vill kunna använda klasserna i `uppgift2.cc`.

Tips: Du kan använda `deque` istället för `vector` för att lagra vilka tåg som är i ett segment. Deque används på samma sätt som vector och har dessutom den användbara funktionen `pop_front` för att låta främsta tåget lämna segmentet.

Vi bedömer din lösning enligt samma punkter som uppgift 1. Det givna huvudprogrammet kommer vid korrekt lösning ge följande utmatningar:

Körexempel 1

```
Kustpilen enters segment Mjölby-Tranås
X2000 enters segment Mjölby-Tranås
X2000 throttled to max 90 km/h
Rocket 1829 enters segment Mjölby-Tranås
Kustpilen must leave segment Mjölby-Tranås before Intercity can enter
Intercity enters segment Mjölby-Tranås
Intercity throttled to max 48 km/h
```

[1] En varumärkeskonsult rekommenderade redan i bolagets begynnelse att korta akronymen från fyra till tre bokstäver. Ingen minns längre varför.

Uppgift 3 - Påbyggnad för högre betyg

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 `Object`. Implementation av `Object` och exempel på huvudprogram finns i `uppgift3.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 (dv när ruta A och ruta B är samma ruta). Om ruta A är tom flyttas ev objekt i ruta B till ruta A. Om båda rutorna innehåller lika objekt som är på samma nivå så uppgraderas objektet på ruta A en nivå och objektet i ruta B försvinner.
- Samtliga speciella medlemsfunktioner undantaget flytt-tilldelning och kopieringstilldelning som explicit ska tas bort.
- Korrekt minneshantering. Inga minnesläckor tillåts.

Det givna huvudprogrammet kommer vid korrekt implementation skriva ut:

Körexempel 1

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

Uppgift 4 - Påbyggnad för högre betyg

Variabler i programspråket Python är på gott och ont mer flexibla än i C++. I Python behöver vi endast ange variabelns namn och samma variabel kan lagra vilket objekt som helst, vi behöver inte hålla oss till objekt av en förutbestämd typ. Därav är språket mer flexibelt, men det är även mycket svårare att veta om variabeln är en lista, en sträng, ett heltal eller något annat. Det blir då svårt att veta hur den kan används.

I denna uppgift ska du implementera grunden till flexibel variabelhantering i C++. Klassen `Var_Base` fungerar som basklass. Klassen `Number` är en härledd klass för att representera variabler av heltalstyp och ska endast kunna konstrueras utifrån en `int`. Klassen `String` är en härledd klass för att representera variabler av strängtyp och ska endast kunna konstrueras utifrån en `std::string`. Oavsett härledd typ ska det gå att utföra följande operationer på en variabel:

- `A.add(int B)` ska numerisk addera B till A (som A += B) när A representerar ett heltal, och slå samman A med B när A representerar en sträng.
- `A.concat(string B)` ska slå samman A med B när A representerar en sträng, och kasta ett lämpligt undantag när A representerar ett heltal.
- `A.print(OS)` ska skriva ut A på strömmen OS.

För att förenkla användning av de nya klasserna behövs ytterligare en klass `Var` vars syfte är att kunna hålla reda på en variabel oavsett härledd typ. En delvis implementation finns i `uppgift4.cc`. Där finns även ett huvudprogram som ska fungera när allt är klart.
Alla kompileringsvarningar med `w++17` ska korrigeras så som du finner lämpligt och utan att programkörningsfel kan uppstå. Inga minnesfel eller minnesläckor får uppstå.