TDP004 - Tenta

2022-01-13

Regler

- All kod som skickas in för rättning ska kompilera och vara väl testad.
- Inga elektroniska hjälpmedel får medtas. Mobiltelefon ska vara avstängd och ligga i jacka eller väska.
- Inga ytterkläder eller väskor vid skrivplatsen.
- Student får lämna salen tidigast en timme efter tentamens start.
- Vid toalettbesök ska pauslista utanför salen fyllas i.
- All form av kontakt mellan studenter under tentamens gång är strängt förbjuden.
- Böcker och anteckningssidor kan komma att granskas av assistent, vakt eller examinator under tentamens gång.
- Frågor om specifika uppgifter eller om tentamen i stort ska ställas via tentasystemets kommunikationsklient.
- Systemfrågor kan ställas till assistent i sal genom att räcka upp handen.
- Endast uppgifter inskickade före tentamenstidens slut rättas.
- Ingen uppgift kan kompletteras under tentamens sista kvart.
- En uppgift kan som regel kompletteras tills den är antingen "Godkänd" eller "Underkänd". En uppgift bedöms som "Underkänd" om ingen markant förbättring skett sedan tidigare inlämning.
- Kompilerande kod, fullständig kravuppfyllnad och följande av god stil och goda konventioner enligt god programmeringssed är krav för att en uppgift ska bedömas "Godkänd".

Hjälpmedel	En C++-bok (t.ex. Stroustrup) Ett A4-ark med egna anteckningar

Information

Betygssättning vid tentamen

Tentamen består av ett antal uppgifter på varierande nivå. Uppgifter som uppfyller specifikationen samt följer god sed och konventioner ges omdömet "Godkänd". Annars ges omdömet "Kompletteras" eller "Underkänd". Tentamen kräver två godkända uppgifter för betyg 3. Alla betygsgränser ses i tabell 1 och 2. *För betyg 3 har du alltid hela tentamenstiden, varken mer eller mindre.* (För student som från LiU fått rätt till förlängd skrivtid förlängs betygsgränserna i proportion till den förlängda skrivtiden.)

Tid	Lösta uppgifter	Betyg	Tillgodo
4 h	3	5	inget
2.5 h	2	4	inget
4 h	2	3	inget
4 timmar	1	|	löst uppgift

Tabell 1: Betygsättning vid förtentamen (dugga), 4 uppgifter ges

Tid	Lösta uppgifter	Betyg
3 h $+B$	3	5
4 h $+B$	4	5
4 h $+B$	3	4
2 h $+B$	2	4
5 timmar	2	3

Tabell 2: Betygsättning vid sluttentamen, 5 uppgifter ges

Bonustid ($+B$)

All bonustid gäller endast under den första ordinarie tentamen i samband med kursen (januari). Varje moment i kursen som ger bonus ger 5 minuter extra tid för högre betyg på sluttentamen, upp till maximalt 45 minuter. Detta är markerat med $+B$ i tabellen där bonus räknas.

Tillgodoräknanden

Tillgodoräknanden gäller endast under den första ordinarie tentamen i samband med kursen. Om du på förtentamen endast lyckas lösa en uppgift kan du tillgodoräkna motsvarande uppgift på sluttentamen (se tabell 1). Du har då bara en uppgift kvar till betyg 3. För högre betyg (om du löser mer än en uppgift på förtentamen) kan du inte tillgodoräkna något. Om du på sluttentamen löser en andra uppgift snabbt och vill sikta på högre betyg kan du lösa den tillgodoräknade uppgiften "igen", men den räknas endast mot högre betyg, **inte** som andrauppgift för betyg 3.

Inloggning

Logga in på datorn i labbsalen med ditt liu-id och lösenord. Detta är samma inloggningsuppgifter som du använder i Lisam.

Skrivbordsmiljön

När du kommit in i tentasystemet har du en normal skrivbordsmiljö (Mate-session i Ubuntu). Efter en stund kommer även din kommunikationsklient att dyka upp automatiskt. Startmenyn

är nedskalad till enbart det som examinator bedömt relevant. Andra program kan fortfarande finnas tillgängliga genom att starta dem från en terminal. Observera att en del program som använder nätverkstjänster inte fungerar normalt, eftersom nätverket inte är åtkomligt.

När du är inloggad är det viktigt att du har tentaklienten igång hela tiden. Om den inte dykt upp fem minuter efter inloggning och inte heller när du startar den manuellt från menyn (fisken) tar du kontakt med assistent eller vakt i sal.

Terminalkommandon

`e++17` används för att kompilera med "alla" varningar *som fel*.
`w++17` används för att kompilera med "alla" varningar. **Rekommenderas.**
`g++17` används för att kompilera `utan` varningar.
`valgrind --tool=memcheck` används för att leta minnesläckor.

C++ referenssidor

På tentan har du tillgång till referenssidorna på `http://www.cppreference.com/` via webbläsaren Chrome. Starta `chromium-browser` i terminalen eller välj lämpligt alternativ från startmenyn. Observera att allt utom referenssidorna är avstängt. Om du inte kan komma åt en sida du tycker hör till referenssidorna (som kanske blockerats av misstag) kan du skicka ett meddelande via tentaklienten. Tag hjälp av assistent i sal om det inte fungerar.

Givna filer

Eventuella givna filer finns i katalogen `given_files`. Denna underkatalog är skrivskyddad, så det är ingen risk du råkar ändra på dessa filer. Skrivskyddet gör dock att du måste kopiera in de givna filer du vill använda till tentakontots hemkatalog. Hur du listar och kopierar filer ska du kunna. Hemkatalogen står du i från början, och du kommer alltid tillbaka till den genom att bara exekvera kommandot `cd` i terminalen.

Avslutning

När dina uppgiftsbetyg och ditt slutbetyg i kommunikationsklienten stämmer överens med det du förväntar och du är nöjd, eller när tentamenstiden är slut, är det dags att logga ut. Hinner du inte se ditt betyg får du höra av dig till examinator via epost efter tentamen.

Avsluta alla öppna program och logga ut ur datorn. Lämna inte datorn innan du ser att du är utloggad.

Uppgift 1 - Klass, behållare och felhantering

En IPv4-adress består av 4 stycken oktetter. En oktett är bara 8 bitar som normalt representerar ett heltal i interallet 0 till och med 255. Din uppgift är att skapa 2 klasser som tillsammans kan användas för att representera en sådan adress. Klassen `Adress` ska lagra 4 stycken objekt av typen `Octet` i en lämplig behållare. En `Octet` lagrar ett heltal i intervallet 0 till och med 255.

Eftersom en oktett bara ska lagra tal i intervallet 0 till och med 255 så är det viktigt att det finns felkontroll för detta ifall användaren skulle försöka skapa konstiga adresser. Vid felaktiga tal ska ett lämpligt fel kastas. Studera körexemplet i kombination med det givna huvudprogrammet.

Klasserna ska ha följande funktionalitet:
Adress:

- En konstruktor som inte tar några parametrar och representerar en adress som motsvarar 000.000.000.000
- En konstruktor som tar in 4 heltal som sammantaget representerar adressen
- En medlemsfunktion `print` som skriver ut adressen. Varje `Octet` ska skrivas ut separerat med punkt.

Octet:

- En konstruktor som tar ett heltal som parameter
- En medlemsfunktion `print` som skriver ut talet som lagras i objektet. Oktetten skall alltid skrivas ut med alla 3 siffor. Om oktetten lagrar talet 1 skall alltså 001 skrivas ut.

Krav: IPv4-adresser kanske inte räcker i framtiden, så din lösning ska utan större svårighet kunna utökas till att lagrar fler oktetter. Det är alltså inte ok att använda fullständig uppräkning utom i konstruktorn som ska ta 4 heltal.

Körexempel:

```
000.000.000.000
192.016.000.001
Octet outside of range: 0 to 255
Octet outside of range: 0 to 255
```

Uppgift 2 - Klass och operatoröverlagring

Kapten Picard gillar gränssnitt. Operatorer kan användas för att skapa bra gänssnitt till klasser i c++. I c++ är klassen `std::string` mycket användbar men beter sig kanske inte exakt som Picard vill. I denna uppgift ska du skapa en ny klass som heter `Text` som lagrar en datamedlem av typen `std::string` och implementerar ett antal konstruktorer och operatoröverlagringar.

Konstruktorer:

- En konstruktor som inte tar några parametrar och sätter datamedlemmen till en tom sträng
- En konstruktor som tar en sträng som parameter och sätter datamedlemmen till den parametern

Operatorer:

- Operatorn + där vänstra operanden är av typen `Text` och högra operanden är av typen `Text`, som skapar en ny `Text` som är sammanslagningen av operanderna, se körexemplet.
- Operatorn * där vänstra operanden är av typen `Text` och högra operanden är av typen `unsigned int`, som skapar en ny `Text`, den nya texten är texten i den vänstra operanden upprepad lika många gånger som heltalet i den högra operanden, se körexemplet.
- Operatorn - där vänstra operanden är av typen `Text` och högra operanden är av typen `char`, som skapar en ny `Text` där det första tecknet som matchar den vänstra operanden är borttagen, se körexemplet.
- Operatorn « som skriver ut en strängrepresentation av texten till godtycklig utström, se körexemplet.

Körexempel:

```
Hej + Svej = HejSvej
HejSvej * 4 = HejSvejHejSvejHejSvejHejSvej
HejSvej - 'e' = HjSvej
```

Uppgift 3 - STL

Krav: Denna uppgift skall lösas med algoritmer från standardbiblioteket så långt som möjligt. Max ett av stegen i punktlistan nedan får lösas med algoritmen `for_each` **ELLER** en loop.

Major Kiera är intresserad av politik i Sverige. När valresultaten är färdiga så delas mandat ut till partier i Sverige. Det finns ett bestämt antal platser i riksdagen men det är inte säkert att dessa kan delas ut helt rättvist till partierna. I filen `valresultat.txt` finns antalet röster för de olika partierna i ett val. Ditt jobb är att göra följande:

- Läs in filen i en lämplig behållare
- Sortera behållaren i fallande ordning
- Summera alla rösterna
- Räkna om varje partis röster till procent av det totala antalet (double räcker som precision)
- Räkna fram hur många mandat varje parti ska ha, avrundat nedåt
- De överblivna mandaten skall fördelas till partierna i fallande ordning (största partierna får de extra mandaten). **tips:** man kan stega fram en iterator ett antal steg med `std::next`
- Skriv ut hur många mandat varje parti fick samt det totala antalen mandat.

Körexempel:

```
144
100
89
5
4
3
2
2
0
totalt antal mandat: 349
```

I verkligheten är det här ett svårt problem att lösa rättvist. Det finns olika sätt att dela ut de överblivna mandaten. Sättet som beskrivs ovan är alltså en förenkling.

Uppgift4 - Polymorfi

Sisko älskar objektorientering och geometri. Han vill därför att du ska skapa en polymorfisk klasshierarki som representerar geometriska figurer. Du ska implementera följande tre klasser:

Figure

- `Figure` behöver lagra ett namn på figuren i from av en sträng
- Objekt av typen `Figure` skapas genom att användaren skickar in en sträng
- Alla figurer kan beräkna figurens area, det finns inget definerat beteende för detta i denna klass
- `Figure` ska ha en funktion `to_string` som returnerar en strängrepresentation av figuren. Det som är gemensamt för alla figurer skall ingå i denna strängrepresentation.

Sphere

- Objekt av denna typ skapas genom att användaren anger ett namn och radien
- En sfärs volym beräknas genom formeln `(4 * r * r * r) / 3`
- För en sfär utökas också beteendet för utskrift, se körexempel och given fil.

Pyramid

- Objekt av denna typ skapas genom att användaren anger ett namn, höjd, djup och bredd.
- En pyramids volym beräknas genom formeln `(h * d * b) / 3`
- För en pyramid utökas också beteendet för utskrift, se körexempel och given fil.

Krav: Koduppprepning ska undvikas
Krav: Eventuella problem med slicing och minnesläckor ska lösas i det givna huvudprogrammet

Körexempel:

```
Figure Pyramids volume is 21.3333
The dimensions are:
height = 4
width = 4
breadth = 4

Figure Spheres volume is 85.3333
The dimensions are:
radius = 4
```

Uppgift5 - Pekare och dynamiskt minne

I denna uppgift ska du skapa en klass, `Upp` (Unique pair pointer), som representerar två unika pekare till heltal. Eftersom pekarna är unika är det av största vikt att objekt av denna typ aldrig kan kopieras, de kan däremot flyttas. Klassen ska ha 2 datamedlemmar av typen `int*` och minst en konstruktor som tar 2 heltal, allokerar dessa i dynamiskt minne (på `heapen`), och sätter datamedlemmarna så att de pekar till det allokerade minnet. Du behöver också se till att klassen ansvarar för att lämna tillbaka det allokerade minnet. Utöver detta behöver du skapa 2 medlemsfunktioner som tillåter åtkomst till de allokerade resurserna på ett lämpligt sätt.

Skapa också ett lämpligt huvudprogram som testar att klassen fungerar som förväntat och inte läcker minne.

Tips: Du kan testa minnesläckor med kommandot `valgrind --leak-check=full ./a.out` (eller dylikt), valgrind kan dock bara upptäcka alla minnesläckor om ditt huvudprogram testar klassen utförligt.