

TDP004 - Tentamen

2016-01-13

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 eller rökpaus 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. C++ Primer 5th ed.) En A4-sida med egna anteckningar
------------	---

Information

Betygsättning 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 tentamens tiden, 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, 4 uppgifter ges

Tid	Lösta uppgifter	Betyg
3 h + <i>B</i>	3	5
4 h + <i>B</i>	4	5
4 h + <i>B</i>	3	4
2 h + <i>B</i>	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

Innan du loggar in ska du välja “Exam system” från sessionsmenyn i nedre vänstra hörnet av inloggningsrutan. Därpå loggar du in med dina LiU inloggningsuppgifter. Du kommer nu in i tentainloggningen och ska börja med att välja språk. Ta den flagga som ser minst Engelsk ut så blir det Svenska (valet spelar ingen roll för dig som kan båda språken). Följ instruktionerna på skärmen så långt det går tills du ska mata in ett engångslösenord. Tag fram ditt LiU-kort och visa det för assistent eller vakt i sal för att få detta lösenord.

Skrivbordsmiljön

När du kommit in i tentasystemet har du en normal skrivbordsmiljö (Mate-session i Linux Mint) med grön skrivbordsbakgrund. 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++11` används för att kompilera med “alla” varningar *som fel*.
`w++11` används för att kompilera med “alla” varningar. **Rekommenderas.**
`g++11` används för att kompilera **utan** varningar.
`gccfilter e++11` används för att köra `e++11` genom `gccfilter`.
`gccfilter w++11` används för att köra `w++11` genom `gccfilter`.
`gccfilter g++11` används för att köra `g++11` genom `gccfilter`.
`valgrind --tool=memcheck` används för att leta minnesläckor.

C++ referenssidor

På tentan har du *experimentiell* tillgång till referenssidorna på <http://www.cplusplus.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. Då lösningen är på experimentell nivå kan det hända att problem uppstår, t.ex. att proxyn inte går att nå. Tag då hjälp av assistent i sal.

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. Hemkatalogen heter alltid `/home/student_tilde` om du undrar.

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 tryck på knappen märkt “Exit” i menyn längst ner på skärmen och välj “ok”. Vänta ett tag och tryck sedan på knappen “Avsluta tentamen” när det är möjligt. När detta är gjort är det omöjligt att logga in igen. Lämna inte datorn förrän du ser den vanliga inloggningsskärmen med blå bakgrund. Anmäl till assistent eller vakt om inloggningsskärmen inte dyker upp inom en minut så åtgärdar vi problemet.

Uppgift 1 Stapla klot

Inledning

Vi tänker oss N stycken genomskinliga glasrör ståendes på rad. Som ett spel kan man tänka sig att vi släpper ner klot (orbs) från ovan och låter dem landa i rören. Vi tänker oss att det är slumpmässigt vilket rör som just varje klot hamnar i. Vi antar även att rören är väldigt långa och smala och att det kan finnas maximalt 99 stycken rör.

Funktionella krav

Skriv ett program där användaren får mata in N , d.v.s. hur många rör vi har. Användaren skall därefter mata in hur många klot vi vill släppa ned (också maximalt 99 stycken). Programmet skall därefter slumpa hur kloten hamnar i rören och rita ut detta på skärmen. I den utritade "bilden" skall även rören vara numrerade från 1 upp till N . Ingen felhantering av indata krävs.

Ickefunktionella krav

- Du ska generera slumpstal enligt C++11-standard.
- Din lösning ska använda minst en STL-behållare och minst en STL-algoritm.
- Din lösning ska ha minst en lämplig funktion utöver `main`.
- Utdata ska matcha körexemplen till formatet (inte exakt eftersom data är slumpat).

Körexempel (användarinmatning i fet stil)

Enter N: **5**

Enter orb count: **13**

```

      o
    o  o
  o  o o
o  o o o
1 2 3 4 5
```

Enter N: **4**

Enter orb count: **7**

```

      o
    o o
  o o o o
 1 2 3 4
```

Enter N: **13**

Enter orb count: **7**

```

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

Uppgift 2 Nixlista

Inledning

När det ringer är det då och då ett samtal man inte vill ha och vill blockera för all framtid (t.ex. försäljare). På filen `given_files/do_not_answer.txt` finns telefonnummer som ska blockeras. Det är givet att det ligger exakt ett telefonnummer per rad i filen. Ett inrikes telefonnummer kan innehålla enbart siffror och bindestreck. Ett utrikes nummer inleds dessutom med ett plustecken direkt följt av två landssiffror. Dessutom kan siffrorna vara grupperade med blanksteg. T.ex. är både 013-28 21 46 och +46 13 282 146 giltigt format (och samma nummer).

Funktionella krav

Din uppgift är att skapa ett program som ber användaren mata in telefonnummer. Efter varje inmatning ska programmet skriva ut om numret är blockerat eller ej. Programmet ska avslutas när användaren trycker Ctrl-D på tom rad.

Ett problem är att användaren inte alltid matar in ett telefonnummer på samma format som det som eventuellt finns i filen. Ditt program ska se till att telefonnumren ändå matchar genom att jämföra att det är samma siffror i rätt ordning. Ett inrikes nummer jämförs med ett utrikes nummer genom att bortse från landskoden (många utrikes kan matcha ett inrikes).

Ickefunktionella krav

1. Du ska använda en väl inkapslad klass för att representera ett telefonnummer. Klassen ska ha en medlemsfunktion eller operator för att kunna jämföra med ett annat telefonnummer.
2. Filen får endast läsas igenom en gång.
3. Korrekt filseparering och inkluderingsgard krävs.
4. Utdata ska matcha körexemplen till fullo.

Tips

Det är okej att rensa telefonnummer från allt som inte är siffror bara det sker i den kod som bör ha det ansvaret och du fortfarande hanterar landskoder korrekt.

Körexempel (användarinmatning i fet stil)

Enter one phone number per line (finish by Ctrl-D):

08-768 43 55

Blocked.

02005301821

Blocked.

+46 070 787-35 17

Blocked.

+48 070-78 73 517

Blocked.

013 14 16 22

This number might be okay

Uppgift 3 Produktsummering

Inledning

I `given_files/order_book.txt` finns en samling produktbeställningar. En rad i filen innehåller först ett produktnummer (1-7 siffror) och sedan ett antal. Alla raderna i filen har korrekt format.

Funktionella krav

Ett problem är att en sådan här fil kan innehålla många beställningar, så att samma produktnummer kan finnas på flera rader. Din uppgift är att skapa ett program som går igenom angiven fil gång och skriver ut totalsumman beställningar för varje unikt produktnummer. Du måste alltså summera antalet för alla lika produktnummer, se exempel.

Utskriften ska vara i form av en snygg tabell sorterad enligt produktnummer, lägst först. Produktnumret ska vara vänsterställt och antalet högerställt.

Ickefunktionella krav

1. Namnet på filen med beställningar ska ges på kommandoraden, med alla tillämpliga felkontroller.
2. Lösningen ska använda en lämplig STL-container.
3. Filen får endast läsas igenom en gång.
4. Utdata ska matcha körexemplen till fullo.

Körexempel (användarinmatning i fet stil)

```
$ a.out
```

```
Usage: a.out FILE
```

```
$ ./a.out given_files/order_book.txt and-another-arg
```

```
Usage: ./a.out FILE
```

```
$ ./a.out given_files/no_such_file.txt
```

```
Error: 'given_files/no_such_file.txt' can not be opened!
```

```
$ ./a.out given_files/order_book.txt
```

```
PID      AMOUNT
13        123
16        164
22         16
31         25
```

Uppgift 4 Syldaviskt krypto

Inledning

Den syldaviska armen har en ny kod som de använder för att säkert kommunicera mellan sina regimenten. Koden fungerar på följande sätt:

För alla bokstäver mellan 'A' och 'Z' och mellan 'a' och 'z' så vänder man på alfabetet. 'a' blir alltså 'z' och vice versa, 'b' blir 'y' och vice versa o.s.v. Alla tecken som inte är bokstäver är helt oförändrade. T.ex. blir strängen "hejsan" "svqhzm" och "Anfall nu!" blir "Zmuzoo mf!". Det som är så enkelt med denna kod är att avkrypteringen kan ske med samma algoritm som krypteringen!

Funktionella krav

Skriv ett program som låter användaren mata in ett meddelande (på en rad) och som sedan skriver ut det krypterade meddelandet. *Beakta noga de icke-funktionella kraven!*

Ickefunktionella krav

Tids nog kommer syldaviska armen att vilja byta krypto. Då är det bra om koden är väl förberedd för detta. Därför ska du skriva en funktion för koden som läser in ett meddelande och skriver ut den krypterade versionen. Funktionen ska ta in en konstant kryptoklassreferens som enda parameter. Kryptoklassen ska vara abstrakt med medlemsfunktionerna **encrypt** och **decrypt**. I huvudprogrammet skapas en subclass till kryptoklassen som skickas som argument till funktionen. Subklassen implementerar det krypto som ska användas. Du ska skapa två olika subclasser, en som implementerar "ingen kryptering" och en som implementerar syldaviska armens krypto.

Båda kryptosubklasserna ska fungera att använda och den Syldaviska ska vara aktiv vid inlämning.

Körexempel 1 (användarinmatning i fet stil)

Enter message: **No one will crack this code!**

Crypto message: Ml lmv droo xizxp gsrh xlwv!

Körexempel 2 (användarinmatning i fet stil)

Enter message: **Roses are red, violets blue.**

Crypto message: Ilhvh ziv ivw, erlovgh yofv.

Uppgift 5 Stack

Inledning

En stack börjar tom. Nya värden läggs alltid till sist, ett värde i taget, men till skillnad från en kö plockas värden även ut från slutet. Värden kommer alltså ut i omvänd ordning mot vad de stoppades in. Detta brukar även kallas LIFO (Last In, First Out). Eller mer sällsynt LCFS (Last Come, First Served).

Funktionella krav

Skriv en klass som representerar en stack med operationerna för att lägga till en sträng i stacken, plocka ut en sträng från stacken, och kontrollera om stacken är tom.

Ickefunktionella krav

1. Din stack ska byggas upp av stackelement (barr) där vardera barr pekar på nästa barr.
2. Din stack ska ha korrekt minneshantering.
3. Din stack ska använda god abstraktion och inkapsling.
4. Operationerna på stacken ska heta *push* för insättning, *pop* för borttagning och *empty* för att kontrollera om stacken är tom.
5. Nyckelordet `const` ska användas där det är lämpligt.
6. Din klass ska skrivas med korrekt inkluderings- och implementationsfil.
7. Kopieringskonstruktor, tilldelningsoperator, movekonstruktor och movetilldelning ska vara korrekt implementerade eller åtgärdade på sådant sätt att användning av dem ger kompilersfel.
8. Det givna huvudprogrammet ska fungera utan modifikation.
9. Utskrifterna ska matcha körexemplen till fullo.

Tips och begränsningar

- Det är lämpligt att ordna stacken på sådant sätt att insättning kan ske på enkelt sätt utan att iterera genom hela stacken.
- Filen `given_files/stack.h` innehåller en början på klassdefinition som dock behöver kompletteras och modifieras för att uppfylla vissa av ovan krav.
- Tänk på att skicka med alla filer som behövs vid kompilering när du är klar.
- Det är inte krav på undantagshantering, men utskriften ska matcha körexemplen till fullo.

Körexempel (användarinmatning kursivt, filslut fetstilt)

```
empty stack
testing 1 2 3
Ctrl-D
3 2 1 testing empty stack
empty stack
```