TDP004 - Tentamen

2014-01-15

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 tentamnenstidens 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

Betygssättning vid tentamen

Tentamen består av fem uppgifter på varierande nivå. Uppgifter som uppfyller specifikationen och 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. *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
3 h ±	3	5
4 h ±	4	5
4 h ±	3	4
5 timmar	2	3

Tabell 1: Betygsättning vid tentamen

Bonus och tillgodoräknanden vid första tentamen (±)

All bonus och alla tillgodoräknanden gäller endast under den första ordinarie tentamen i samband med kursen. Varje avklarad deadline i kursen ger bonus i form av 5 minuter extra tid för betyg 4 och 5, upp till maximalt 35 minuter.

Varje "Godkänd" uppgift på duggan ger en motsvarande tillgodoräknad uppgift på tentamen. Tidsåtgången för en tillgodoräknad uppgift sätts på tentamen enligt schablon till 45 minuter, eller verkliga tidsåtgången på tentamen om den är bättre (kortare).

Om du på duggan löser två uppgifter på 1 timme räknas det alltså som att dessa tagit 1 timme på tentamen, men tar det 2.5 timmar att lösa på duggan räknas det som att dessa "bara" tagit 1.5 timme på tentamen.

Inloggning

Logga in på tentamenskontot med följande användaruppgifter:

```
Användarnamn: examx
Lösenord:     kluring1
```

Följ menyvalen 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.

Skrivbordsmenyn

Genom att högerklicka på skrivbordet (bakgrunden) får du fram skrivbordsmenyn. Där finns alternativ för att starta tentaklienten, en terminal, emacs och (som experimentell beta) textbaserad tillgång till cplusplus.com referenssidor.

När du är inloggad är det viktigt att du direkt startar tentaklienten genom att högerklicka på bakgrunden och välja "tentaklient" i menyn.

Terminalkommandon

w++11 används för att kompilera med "alla" varningar.

g++11 används för att kompilera utan varningar.

w++11filter används för att köra *w++11* med gccfilter.

g++11filter används för att köra *g++11* med gccfilter.

bcheck ./a.out används för att leta minnesläckor.

cppreference startar en terminal med textbaserad tillgång till cplusplus.com referenssidor.

C++ referenssidor

Detta är en "experimentell beta" som provades första gången på duggan i TDP004 2013-12-09 och ger textbaserad tillgång till C++ referenssidor på cplusplus.com. Det går bara att nå referenssidorna på cplusplus.com, forumsidorna liksom resten av internet är avstängt. Om det inte fungerar hänvisas du till boken.

Använd piltangenterna och enter för att navigera. Numeriska tangentbordets "Page up" och "Page down" scrollar upp eller ned (toggla "Numlock" om det inte fungerar). Tryck "Esc" för att få fram en meny, och "/" för att söka på en sida. Mer kommandon finns i hälpmenyn under alternativet "Keys".

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

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 Min-generator

Denna uppgift tillgodoräknas för dig som löst uppgift 1 på duggan. Detta kommer visas i din tentamensklient en tid efter tentamens start.

Inledning

Spelet "MS röj" är en gammal klassiker för de som kör Windows. Spelet består av ett rutnät med en viss bredd och höjd. I rutnätet är ett bestämt antal minor slumpvis utspridda. Rutor utan minor innehåller en siffra som anger hur många minor det finns på de åtta omgivande rutorna.

Funktionella krav

Du ska skriva ett program som slumpar fram var i spelet minorna skall hamna. Programmet skall vid start fråga efter spelplanens bredd, höjd och antal minor som skall placeras (användaren matar in korrekta data). Därefter skall spelplanen ritas ut med minor slumpvis utplacerade. En ruta utan mina ritas som en punkt ('.'), medan en mina ritas som en stjärna ('*').

Ickefunktionella krav

- Koordinaterna för varje mina skall slumpas fram innan spelplanen ritas ut.
- Programmet skall alltid placera exakt så många minor som begärts. Det går inte att placera mer än en mina per ruta.
- Varje ruta skall ha samma sannolikhet att få en mina.
- Utskrifterna ska matcha körexemplen till fullo, bortsett från att minorna (stjärnorna) skall hamna på olika plats från körning till körning när de slumpats korrekt.

Körexempel

```
Mata in spelplanens bredd: 10
Mata in spelplanens höjd : 10
Mata in antalet minor    : 10

..........
..........
..*....*..
..........
.*..*....*
.*.......*
....*.....
.......*..
.......*..
..........
```

Uppgift 2 Långa filmer (given_files/moviedb_*.txt)

Denna uppgift tillgodoräknas för dig som löst uppgift 2 på duggan.

Inledning

Två av dina vänner inleder en diskussion om filmers längd genom tiderna. Den ena säger bestämt att filmer genom åren har blivit längre och längre, och som exempel tar hen storfilmer som "Sagan om Ringen" och "Avatar". Den andra hävdar att detta är rent nys, långfilmer har alltid varit långa. Diskussionen blir allt hetare tills du tröttnar och bestämmer dig för att avgöra saken en gång för alla. Du webscrapar helt enkelt imdb.com och sparar årtal, filmens längd i minuter, och filmens titel i en enkel textfil. Filerna innehåller bara amerikanska långfilmer från åren 1964-2013. Utvalda filmer kvalar dessutom in bland de 1000 bästa respektive sämsta.

Funktionella krav

Därpå skriver du ett program som läser textfilen och skriver ut filmernas medellängd genom åren. Programmet startar med att fråga efter ett filnamn med filminformation och beräknar därefter statistiken som slutligen skrivs ut. Om filen inte kan öppnas skall programmet avsluta med ett felmeddelande.

Ickefunktionella krav

- Filmernas medellängd beräknas i femårsintervall med start på 1960.
- Skapa en klass "Average" som lagrar dels en summa och dels hur många värden som summerats. Klassen skall ha en lämplig konstruktor, en funktion som lägger till ett värde till summan, och en funktion för att hämta ut medelvärdet. Använd klassen för att stegvis lägga till filmer till en femårsperiod, och hämta ut medelvärdet när allt är klart.
- Utskrifterna ska matcha körexemplen till fullo.

Tips och begränsningar

- Det är lämpligt att använda datatypen std::map<Five_Year, Average>.
- Klassen "Five_Year" är given i "given_files/movielength.cc".
- Det är okej att skriva hela programmet i en och samma fil.

Körexempel

```
Enter movie information file: moviedb_bottom_1000.txt
1960-1964 : 79.8
1965-1969 : 81.9
1970-1974 : 95.0
1975-1979 : 93.9
1980-1984 : 97.3
1985-1989 : 90.4
1990-1994 : 93.3
1995-1999 : 92.7
2000-2004 : 92.0
2005-2009 : 92.7
2010-2014 : 88.6
```

Uppgift 3 Cirkulärt länkad lista (given_files/circular.cc)

Inledning

I vissa situationer finns en turordning som upprepas om igen i oändlighet (eller kortare). Ett exempel är turordningen vid brädspel som ju går runt tills spelet är slut. När detta implementeras är det praktiskt att ha en cirkulärt länkad lista. Till skillnad från en rak enkellänkad lista (som ju slutar med en nullpekare) så "slutar" en cirkulärt länkad lista med att peka tillbaka till första elementet. På så sätt har den som cirkeln ingen ände, stegar man igenom listan kommer det bara gå runt runt. Som tur är blir inte datorn yr.

Funktionella krav

Färdigställ den påbörjade cirkulärt enkellänkade listan. Det som saknas är funktionen för insättning, utskrift och destruktorn.

Ickefunktionella krav

- Insättning skall ske först i listan. En tom lista representeras med en nollpekare. I en lista med endast ett element kommer elementet innehålla en pekare till sig självt. I längre listor kommer "sista" elementet innehålla en pekare till "första" elementet.
- Utskriftfunktionen skall skriva ut alla element i listan exakt en gång, med start på det första (senast insatta) elementet.
- Destruktorn skall se till att allt minne avallokeras korrekt. Tänk på att destruktorn för varje element kommer att anropas rekursivt. Det kan vara lämpligt att bryta cirkeln innan destruering.
- Utskrifterna ska matcha körexemplen till fullo.

Tips och begränsningar

- Fundera på vilket element i listan "entry"-pekaren skall peka ut för att insättning först skall bli så enkel som möjligt.
- Det är okej att skriva hela programmet i en och samma fil.

Körexempel

```
Skriv in namn för spelare, avsluta med Ctrl-D.
Namn? a
Listans innehåll:
a->
Namn? b
Listans innehåll:
b->a->
Namn? c
Listans innehåll:
c->b->a->
Namn? <Ctrl-D>
Tjugo stegningar i listan:
c->b->a->c->b->a->c->b->a->c->b->a->c->b->a->c->b->a->c->b->
```

Uppgift 4 Python range (given_files/range.cc)

Inledning

I programspråket python används "range" för att representera en sekvens av tal. Funktionen "range" tar 1, 2 eller 3 heltalsparametrar. Anges en parameter N skall sekvensen från 0 upp till, men inte inklusive N, genereras i steg om 1. Anges två parametrar S och N börjar sekvensen istället på värdet av första parametern S. Anges en tredje parameter L skall sekvensen gå i steg om L istället för i steg om 1, och det gäller då att stoppa innan slutvärdet N. Om steglängden är negativ gäller att N måste komma före S, annars genereras en tom sekvens.

Funktionella krav

Skriv funktionen range i C++ så att den fungerar som i python och kan användas enligt exemplen i "given_files/range.cc". Programmet skall ta in en, två eller tre parametrar via kommandoraden. Dessa skall passas vidare till din range-funktion. Om kommandoraden inte innehåller ett, två, eller tre *heltal* avslutas programmet med ett felmeddelande.

Ickefunktionella krav

- Du får inte skriva mer än två funktioner med namnet "range" och den ena får inte bestå av mer än en programsats.
- Du skall i detta fall testa alla tre sätt att anropa funktionen, och det är därför okej att räkna upp tre stycken med likartad kod.
- Utskrifterna ska matcha körexemplen till fullo.

Tips och begränsningar

- Funktionsöverlagring och defaultvärden är bra att känna till.
- Du kan lösa problemet på ett enklelt men ineffektivt sätt. Det går (svårt) att lösa problemet på rätt sätt (bra och effektivt) genom att skapa en klass enligt iteratorkoncept.

Körexempel

```
$ w++11 range.cc -o range
$ ./range -10
[ ]
$ ./range 10
[ 0 1 2 3 4 5 6 7 8 9 ]
$ ./range 4 10
[ 4 5 6 7 8 9 ]
$ ./range 4 10 3
[ 4 7 ]
$ ./range 4 10 -2
[ ]
$ ./range 4 -10 -2
[ 4 2 0 -2 -4 -6 -8 ]
$ ./range -4 10
[ -4 -3 -2 -1 0 1 2 3 4 5 6 7 8 9 ]
$ ./range -4 10 -1
[ ]
```

Uppgift 5 Kryssfrågor (given_files/quizcheck.h,o)

Inledning

Ibland kan man stöta på en tentamen med kryssfrågor. Denna fråga är en sådan. Det finns 20 frågor med 5 alternativ vardera. Det gäller att kryssa för samtliga korrekta alternativ. En fråga kan alltså besvaras med 0 till 5 kryss. Totalt blir det 100 kryss på de 20 frågorna, så chansen att gissa rätta raden blir rätt liten, en på 2^{100} (det är ett tal med 31 decimala siffror). I denna uppgift visar det sig att examinatorn har gjort fel, han har råkat ge ut headerfilen för rättningsfunktionen istället för textfilen med uppgifterna. Dessutom har den kompilerade objektfilen tillhörande headerfilen råkat komma med bland de givna filerna.

Funktionella krav

Rättningsfunktionen "check" tar emot en lösningsvektor, d.v.s. en std::vector med 20 strängar. En sträng `-----` motsvarar att inget alternativ är förkryssat, och en sträng `|||||` motsvarar att alla fem alternativ är kryssade. Funktionen returnerar totalt antal poäng som erhålls för given lösningsvektor. Det ges ett poäng per uppgift, om alla kryss på uppgiften sitter rätt.

Skriv ett program som skriver ut den rätta lösningen på de 20 kryssfrågorna. Varje rad skall vara en sträng enligt ovan som motsvarar rätta uppsättningen kryss för motsvarande fråga.

Tips och begränsningar

Det kan vara intressant att i sammanhanget veta hur binär addition med 1 går till. Vi utgår som exempel från binära talet "01011". Vi börjar längst till höger. Ett plus ett blir två. Därmed blir resultatet 0 på positionen och vi måste fortsätta addera 1 till nästa position. Även denna är ett från början, så resultatet på position två blir 0, och vi fortsätter addera 1 till position 3. Position 3 har värdet 0. Resultatet blir nu 1 och vi är klara! Resterande positioner ändras inte, Resultatet är "01100".

Finessen med detta är att om vi upprepat adderar 1 till ett 5-siffrigt binärt tal så kommer talet anta samtliga möjliga (32st) kombinationer av 1 och 0. Kan vi nyttja detta till att ta reda på rätt svar på en kryssfråga i taget så går kryssuppgiften att lösa!

Exemplet nedan har bara löst uppgiften för 10 helt andra kryssfrågor. Det är alltså inte det resultat du kommer få, bara exempel på hur det skulle kunna se ut.

Exempel programutskrift ut vid 10 andra kryssfrågor

```
---|-
-||--
|---|
--|-|
-----
-|---
|||||
-||||
|----
-|-|-
```